Foxboro®

by Schneider Electric

Foxboro Evo™
Process Automation System

Information Network Interface Software User's Guide

B0400BM

Rev D
July 21, 2015

SOFTWARE LICENSE AND COPYRIGHT INFORMATION

Before using the Invensys Systems, Inc. supplied software supported by this documentation, you should read and understand the following information concerning copyrighted software.

1. The license provisions in the software license for your system govern your obligations and usage rights to the software described in this documentation. If any portion of those license provisions is violated, Invensys Systems, Inc. will no longer provide you with support services and assumes no further responsibilities for your system or its operation.
2. All software issued by Invensys Systems, Inc. and copies of the software that you are specifically permitted to make, are protected in accordance with Federal copyright laws. It is illegal to make copies of any software media provided to you by Invensys Systems, Inc. for any purpose other than those purposes mentioned in the software license.

Contents

Figures

Tables

Preface

Purpose

This document explains the purpose of the Information Network Interface (INI) software. This document covers in detail the implementation of data and message transfer between two separate Foxboro Evo™ or I/A Series® systems.

Audience

This document assumes a user who is familiar with:

- The system's Operating System.
- The use of a text editor.
- The configuration of the display manager for remote operation.
- The configuration of the alarm manager.
- The configuration of the Annunciator Keyboards and horns.

If the Message Forwarding Service is to be used, the user needs to be familiar with:

- The Alarm System.
- The configuration of the alarm manager.
- The configuration of the Annunciator Keyboards and horns.

If the Message Relay Service is to be used, the user needs to be familiar with:

- The IPC Communication Mechanism.

Revision Information

The following changes were made to B0400BM, Rev. D.

Global changes made for the support of Foxboro Evo Control Core Services.

Reference Documents

The following documents are referenced in this manual and may be worth reviewing prior to using the INI software.

- *AIM*AT Suite AIM*API User's Guide* (B0193YN)
- *I/A Series FoxAPI™ User's Guide* (B0193UD)
- *I/A Series FoxAPI™ Installation Guide* (B0193UC)
- *Display Engineering for FoxView Software and Display Manager Software* (B0193MQ)
- *Display Manager Commands* (B0193DF)
- *Integrated Control Block Descriptions* (B0193AX)
- *Workstation Alarm Management* (B0193RV)

- *Process Alarm Configuration* (B0193AU)
- *Inter-Process Communications Calls* (B0193BB)
- *The MESH Control Network System Planning and Sizing* (B0700AX)
- *Application Object Services User's Guide* (B0400BZ)

Most of these documents are available on the Foxboro Evo Electronic Documentation media (K0174MA). The latest revisions of each document are also available through our Invensys Global Support at http://support.ips.invensys.com.

Terms and Definitions

The following is a list of terms and definitions as they pertain to the INI product.

AIM*Historian
: The AIM*Historian™ database.

Application
: A collection of Application Objects. Generally, an Application (or several) is associated with a program that populates some of the Application Object Attributes and consumes data from others. If the data transfer services are used, the program can use the Application Object Attributes to obtain process data and to move data into the process.

Application Definition
: An Application Definition is a set of records in the database that defines a particular Application. An up to date map file is considered by some to be the Application Definition, but this is incorrect. The true Application Definition is the information about the Application in the database.

Application Object Alarming (aoAlm)
: An optional service that supports the generation of process alarm messages based on the values in the Application Object Attributes.

Application Object Attribute (A:O.A or AOA)
: An attribute of an Application Object. The attributes of an AO hold the value and status associated with a name.

Application Object Services (AOS)
: AOS provides a set of services to users of Application Objects. These services include creation, deletion, mapping, checkpointing, and alarming.

Application Objects (AOs)
: Applications Objects is the generic term for a type of OM variable object. The term is used to refer to:

  All Application Object Attributes collectively.

  The collection of Application Object Attributes belonging to a particular second level name.

Application Objects reside in a workstation and are hierarchical like the objects found in a Control Station. The hierarchy is: Application (A), Object (A:O or AO), and Attribute (AOA or A:O.A).

Though their names resemble Control Block Names, AOs do not have the support of a Block Processor.

Application Objects are often used to represent data structures that would normally be internal to a program (application), but which are more useful if they are globally accessible through the OM. FoxBridge uses AOS in this manner.

Compound Block Parameter (CBP or C:B.P)

The name of a value associated with a Control Block.

Control Network — The network that connects control stations. Components of this network may include nodebus segments, Carrierband LAN segments, and/or network segments on the Foxboro Control Network.

control software — This can be either Foxboro Evo Control Core Services or I/A Series software.

control system — This can be either a Foxboro Evo™ or I/A Series system.

Foxboro Evo Control Core Services

Core software environment, formerly know as the “I/A Series (Intelligent Automation Series) software”.

Foxboro Evo Control Core Services workstation

A workstation which runs the Foxboro Evo Control Core Services without the Foxboro Evo Control Software.

Foxboro Evo Control Software

Formerly known as the “Foxboro Control Software (FCS)” and “InFusion”.

Foxboro Evo Control Workstation

A workstation which runs the Foxboro Evo Control Core Services and the Foxboro Evo Control Software.

Customer Supplied Network (CSN)

The network used to link the Local and Remote systems. Typically, the connection between these two systems uses the optional Ethernet ports found on the workstations.

globally known (logical) name

The IPC mechanism is based on communication by logical name. A name can be locally known or globally known.

Locally known names are known only to the IPC service in the workstation running the program. Globally known names are registered

with the Object Manager. The station address of globally known programs can be found using OM calls. Locally known programs cannot be found in that manner.

Programs using IPC to communicate with a locally known program must either know the other program's station address or it must use the broadcast message. The station address of the other program can be obtained from its globally known name if available.

iaToQ — Process on the Remote Station that is responsible for intercepting and collecting the Control Station, System Monitor, and Operator Action Journal messages. It places messages in an outgoing message queue for processing by **qToNet**.

Information Network Interface Software

A suite of applications that loosely couples separate control systems. This loose coupling allows data exchange and message transfer.

INI — Process on the Local Station that arranges the transfer of data between two Foxboro stations. The communication is bi-directional, but the predominant transfer is from the Remote Station to the Local Station; the host of the **INI** process is considered to be a Local Station.

Local Station — The Foxboro station that receives the off-node alarm messages and/or off-node process data. It is this machine that runs the **INI** process and the **netToIA** process.

locally known (logical) name

The IPC mechanism is based on communication by name. A name can be locally known or globally known.

Locally known names are known only to the IPC service in the workstation running the program. Globally known names are registered with the Object Manager. The station address of globally known programs can be found using OM calls. Locally known programs cannot be found in that manner.

Programs written to communication with a locally known program must either know its station address or must use the broadcast message.

Logical Name — A logical name is an ASCII string used to represent and object. An object may be a value, a program, device, or alias. The INI software is concerned only the values and programs. The Object Manager knows globally known logical names like the value of a control block parameter (CBP) and the name of a historian. Locally known names are used only with programs. They are not registered with the OM, but they are registered with the Foxboro® IPC layer.

map File — An ASCII file used to define the Application Object Attributes to be stored in the database used to hold Application definitions.

mapping	The term used for the transfer of data between an Application Object and another OM variable under the control of the AOS/INI mapping service.
msgRelay	The INI software component responsible for relaying Foxboro IPC messages between application programs.
netToIA	Process on the Local Station that is responsible for receiving all messages sent across the TCP/IP connection from **qToNet** and delivering them to their final destination. The final destination could be a printer, historian, etc. on the Local Station.
NetAIM*API	Networked AIM*API.
Object Manager	The Name Server. It provides services that allow programs to locate and access objects. The most common objects are values and programs using Foxboro IPC.
Object Template	A Bourne shell script that defines the attributes of an Application Object. It is combined with the information found in an **aod** file by **MkMapFile** to form a map file.
qToNet	Process on the Remote Station that is responsible for establishing the TCP/IP connection and sending the collected messages to its partner **netToIA**. It reads the messages placed in the outgoing message queue by **iaToQ**.
Remote Station	The Foxboro station on the other side of a customer supplied network. This is the station that has either process data or messages that require forwarding to the local Foxboro Evo/I/A Series System. It is the station that runs the NetFoxAPI/NetAIM*API server in support of the INI software and the **iaToQ** and **qToNet** applications.
workstation	Can be a Foxboro Evo Control Core Services, Foxboro Evo Control Software, or I/A Series workstation.

1. Overview

This document covers the use of the Information Network Interface (INI) software to inter-connect independent Foxboro Evo/I/A Series systems such that inter-operation is easy and transparent.

The INI software provides services that enable data transfer and message forwarding. Other products provide for remote configuration and system management. Combined these products provide a comprehensive solution to the problems posed by the need to link independent systems.

Goal

There are several major reasons for connecting two or more independent Foxboro Evo/I/A Series systems over a customer supplied network:

- The systems are too far apart to be linked by the Control Network.
- The systems must remain independent so that upgrades are easier.
- The systems were built independently and, therefore, have name conflicts, e.g., stations that have the same letterbug or compounds with the same names.
- The systems were built independently and cannot be rebooted to allow them to be connected to a Control Network.

Implementation

The Information Network Interface software forms a part of the solution to this problem. The INI software consists of three modules:

- The Data Transfer facility which makes the Remote Station's objects available to the Local Station.
- The Message Forwarding facility which forwards the Remote Station's alarms and other messages to the Local Station for annunciation.
- The Message Relay facility that supports two-way communication between Foxboro Inter-Process Communication based programs.

The following sections explain the needs of a complete solution, the capabilities of an INI software based solution, the system and network requirements, and the method of operation for all three components of the Information Network Interface package.

Customer Supplied Network (CSN)

The preferred Customer Supplied Network solution for use with the INI software has the following characteristics if the station is to be used as part of the remote access solution:

- The CSN is TCP/IP based.
- The CSN provides common TCP/IP tools.
- The CSN supplies significant bandwidth.
- The CSN supplies a low latency communication path.

If a server is not used, the bandwidth requirement is significantly less and the latency requirement is not as strict.

Equipment Recommendations

The preferred INI solution uses:

- One or more servers at the remote location.
- Terminal Server Edition software is used to access remote displays and tools.
- Optionally, one may run the INI software. This allows the transfer of process data and messages.
- It is highly recommended to use managed switches and network management software to monitor the switch operation.
- The use of fiber optic connections from the workstation or server to the switch is generally preferred since it will eliminate certain problems.

Equipment Settings - NIC to Switch

When a workstation or server is connected to a switch or a Customer Supplied Network, the user needs to ensure that the workstation's Network Interface Card (NIC) is configured to match the settings on the Customer Supplied Network.

While auto-negotiation and auto-sensing are comment facilities, the result can be a combination of settings that have an adverse impact on the INIs operation.

The person responsible for installing the INI software should work with the local network services group to ensure that the settings match. Commonly, the settings would be 100 Mbps, full duplex.

— TIP

Errors that could be encountered would include high levels of CRC, alignments, and collision errors reported on the switch.

Key Features

The key features that a customer might expect in a Customer Supplied Network (CNS) based solution are presented in the following table along with how those features that can be implemented in a CSN based Microsoft® Windows® system. The table makes clear the facilities available over each network and how they are implemented.

Table 1-1. Features Available over the Control Network and over a Customer Supplied Network

Features	Remote Operations featuring Remote Display Access and the Information Network Interface
Redundancy	The INI software does not directly support redundant operation. However, CSNs often do and this lends some resistance to failure to the INI software. In addition, application redundancy for critical data and messages can be implemented.
Real-Time Information Peer-to-Peer Data Transfer	The INI software can connect the points automatically based on its configuration.
Graphic and other Application Support	The INI software represents remote data locally and these local tags are available to all local applications. The INI software representation enables local applications to address these tags using all normal OM mechanisms: Read Lists, Write Lists, One-shot Gets, and One-Shot sets. The server can use the Microsoft's Terminal Server Edition to provide remote access.
Message Forwarding Process Alarms	INI Message Forwarding application.
System Monitor Message	INI Message Forwarding application. Records messages in a remote historian and retrieves them locally. Records messages in the AIM* Historian using a remote data collector.
Operator Action Journal Messages	INI software. Records messages in a remote historian and retrieves them locally. Records messages in the AIM* Historian using a remote data collector.
Process Operation Process Graphics	Local process graphics access the local Application Object that is representing the remote data. This approach allows the same display to be used at both locations and it has normal display update speeds. However, redundancy is limited and there is some delay as the data travels through the INI software. The server can use Microsoft's Terminal Server Edition to provide remote access.
Current Alarm Display/Alarm Manager	INI Message Forwarding to local alarm manager. The INI software fully supports local annunciation of remote alarms. It provides message forwarding which allows local alarm managers and printers to display remote alarms. It can represent the remote block's UNACK parameter that allows local alarm managers to acknowledge remote alarms. The server can use the Microsoft's Terminal Server Edition to provide remote access.

Table 1-1. Features Available over the Control Network and over a Customer Supplied Network (Continued)

Features	Remote Operations featuring Remote Display Access and the Information Network Interface
Annunciator Keyboard Support	INI software. The FoxPanels application may be of use.
Local Horns	INI software.
Programmatic Access	
FoxAPI	NetFoxAPI software. With some minor rework existing applications can be ported from Local FoxAPI to NetFoxAPI client.
AIM*API	NetAIM*API software. With some minor rework existing applications can be ported from Local AIM*API to NetAIM*API client.
Native APIs	OM API: NetFoxAPI/NetAIM*API replaces the native OM API. Historian API: NetFoxAPI/ NetAIM*API replaces native OM and Historian APIs.
File Transfer	Microsoft networking can be used.
Historical Data	
Collection Group Data Reduction Group Data Archive Group Data	AIM*Historian supports remote collectors with local data access. In addition, data can be collected and stored remotely and accessed locally using several Information Suite tools.
Message Data	INI software can forward messages to local system for storage. AIM* can collect messages remotely and Information Suite tools can access the message data. The server's alarm manager allows viewing of alarms in remote historian.
Configuration	
Displays	The FoxDraw™ application can be run locally on a PC and the files may be stored remotely on the target workstation or server if the network is setup properly.
Control Blocks	The server can use the Microsoft's Terminal Server Edition to provide remote access. The FoxCAE™ software can be used over a properly setup TCP/IP network. The IACC can be used to configure either local or remote systems over the CSN.
System	System Configuration can be done on a local PC, but installation requires physical access to the remote stations.
Others	Most configurators are available remotely. The server can use the Microsoft's Terminal Server Edition to provide remote access.

Table 1-1. Features Available over the Control Network and over a Customer Supplied Network (Continued)

Features	Remote Operations featuring Remote Display Access and the Information Network Interface
System Management	The INI software can forward these messages to local printers and programs. The server can use the Microsoft's Terminal Server Edition to provide remote access.

INI Software

The INI package extends the support for remote operation by adding:

- Message Forwarding,
- Message Relay, and
- Local representation of remote data.

The key features of the INI package are:

1. The INI package is a software-only solution for workstations.
2. The workstations must be connected over a CSN as well as to their local Control Network.
3. The minimum required baud rate is usually quite low for data transfers. Message Forwarding and Message Relay applications may raise the minimum bandwidth.
4. The INI package supports data transfer as follows:
   a. The INI software supports Read, Write, Get, and Set access over a TCP/IP network from one workstation to another.
   b. Local Application Objects represent the tags in the Remote Station in the Local Station.
   c. Because local objects reside in their own CSA space while representing the remote data, these local tags may have exactly the same name as the remote points they represent.
5. The INI software can forward IPC messages through a one-way pipe.
   a. Once forwarded, remote alarm messages can be used for local alarm annunciation: CAD display, horns, and lamps.
   b. The INI package supports Alarm Acknowledgment from the Local Station to the Remote Station.
   c. The INI software supports Alarm Annunciation on the Local Station of remote alarms.
6. The INI software can relay IPC messages through a two-way pipe in support of user-written applications. The particular features of this inter-system service are:
   a. Messages can be sent between programs with globally known names.

b. Messages can be sent between programs where one has a locally known name and the other has a globally known name. This can be done if the locally known program initiates the communication.

c. Messages can be sent between programs by the use of broadcast messages.

The INI package is not redundant in the version covered by this document though the non-redundant hosts can be connected to a redundant CSN to provide partial immunity to CSN failures. Application level redundancy for data and messages is possible, see "Data Redundancy" on page 39.

Data Transfer

The INI software provides local representation of remote data by creating local application objects with the same name as the remote points that they represent. The data is transferred from the Remote Station to the Local Station through the use of the networked version of the FoxAPI/AIM*API software. The following figures show an asymmetric configuration performing data transfer.

— NOTE

Both the local and remote workstations must either be using networked FoxAPI or AIM*API. The mixing of networked FoxAPI and AIM*API between the local and remote workstations is not allowed.

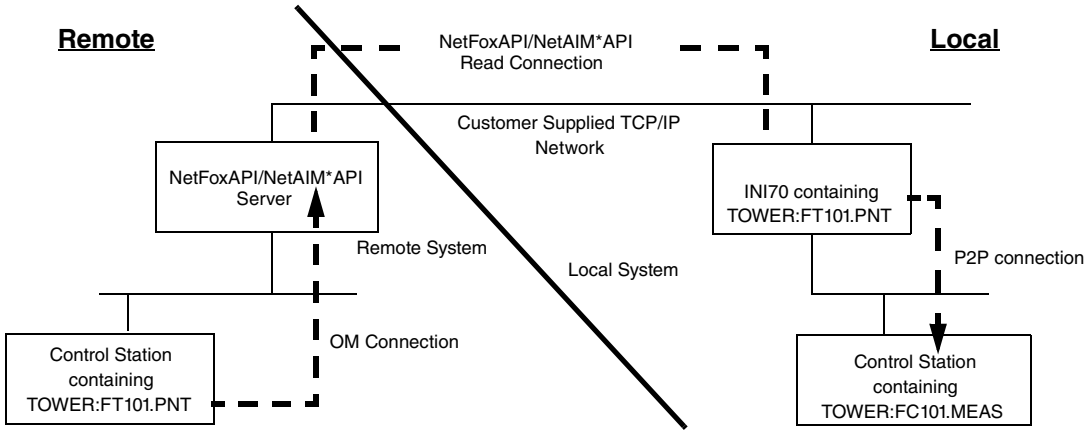


Figure 1-1. Data Flow from Remote Station to Local Station in a Peer-to-Peer Application using an INI R mapping

The INI package fully supports change driven reads of remote data, change driven writes of local data to remote targets, and one-shot sets to remote targets. Local representation of remote data using a read connection provides support for one-shot reads of remote data. The local one-shot read accesses the local representation of the remote data.

— NOTE

Both the local and remote workstations must either be using networked FoxAPI or AIM*API. The mixing of networked FoxAPI and AIM*API between the local and remote workstations is not allowed.

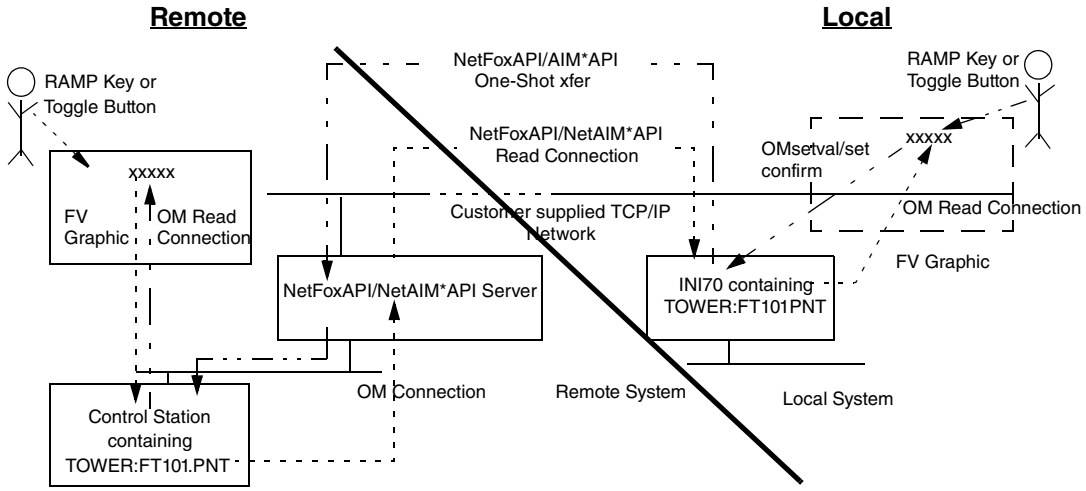


Figure 1-2. Data Flow from Remote Station to Local Station using an INI P Mapping

As figure above shows, the INI software also supports one-shot sets of remote values with read-back. The read-back feature ensures that changes made at either end of the link are reflected at the other end.

Message Forwarding

The INI software provides forwarding of the following message types:

- Control Station Messages
  - Process Alarms
  - Sequence Block Messages generated using **SENDMSG** and **SENDCONF**
  - Sequence of Events Messages generated by the **EVENT** block
- System Monitor Messages
- Operator Action Journal Messages
- Generic messages generated by any user of connectionless IPC messages.

Message Forwarding in the System

Communication within the control network is based on a message passing system built upon a set of communication services known as IPC/COMEX, or the Inter-Process Communication/Executive. IPC/COMEX provides support for transporting a block of data, or packet, from one Foxboro station to another in an efficient and reliable manner.

The Message Forwarding component of the INI package is built on top of these IPC services and allows forwarding of system messages from the remote system to the local system. Message types forwarded include messages generated by Control Stations, System Monitors, and Operator Workstations.

It is important to understand the basics of messaging before attempting to understand the Message Forwarding component of the INI package, please read the appendix on the messages on page 111 to gain this understanding.

Message Forwarding Functionality

The Message Forwarding component of the INI package application is an extension of the Message Handling system. The Message Forwarding component of the INI package application allows messages to be passed to the Alarm Annunciators, printers, and/or historians on a remote control network over a TCP/IP communications link.

Message Forwarding functionality is detailed in the following figure and accompanying text.

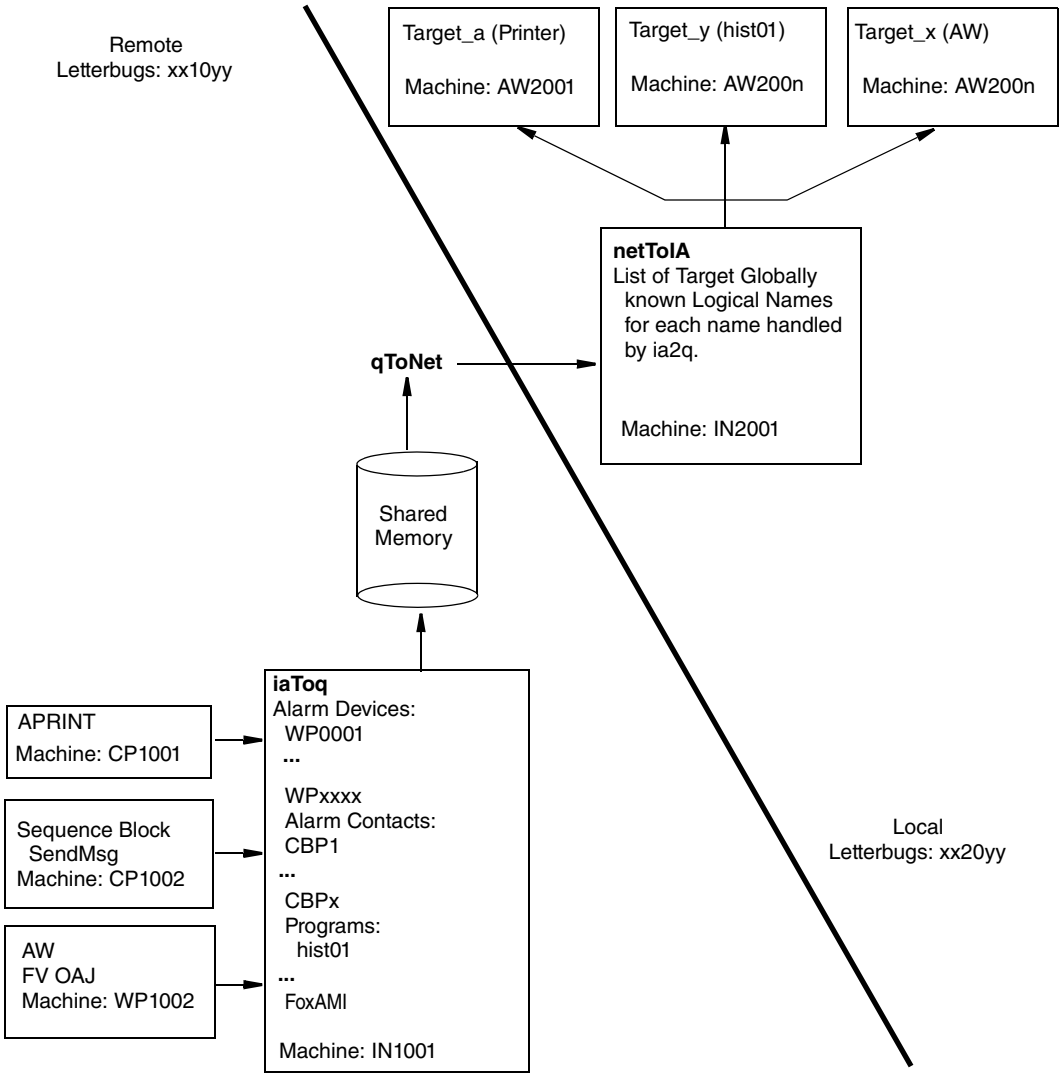


Figure 1-3. Message Forwarding Functionality

The process **iaToQ** runs on the Remote Station and collects the messages that are to be sent to the Local Station.

The process **qToNet** runs on the Remote Station, establishes a TCP/IP connection, and sends the collected messages to the Local Station.

The process **netToIA** runs on the Local Station, receives the messages sent across the TCP/IP connection from Remote Station, delivers them to their final destination. The final destination for the messages could be a printer, historian, or Alarm Alert task in a workstation.

It is important to note that even though Alarm Messages can be sent from a Remote Station to a Local Station, the messages still must be acknowledged on the Remote Station where the messages originated.

If the INI's software data transfer package is used to implement a **P** mapping of the remote block's **UNACK** parameter, the alarms can be acknowledged locally. Otherwise, please review the methods to accomplish remote alarm acknowledgment discussed later in this document.

Message Relay

The INI package can be used as a transparent communications link between programs running on the two systems if the programs use connectionless IPC messages.

The package can:

- Act as a proxy for programs with well-known names, i.e., those registered with the OM.
- Act as a proxy for programs with locally known names communicating to those with a globally known name assuming that the locally known program initiates the communication.
- Relay a broadcast message from one system to the next.

Message Relay and Well-Known Names

A common form of Message Relay involves the transfer of information between two programs whose IPC names are registered with the Object Manager. OM registration of an IPC name allows clients that wish to communicate with the application to find it without knowing the name of the station hosting the application.

For example, the AIM*Historian software uses this facility to receive Process Alarm messages from Control Stations. Each AIM*Historian instance has a globally known name that may be used by Control Stations as an alarm destination.

The INI software fully supports this style of operation. Each **msgRelay** instance can act as a proxy for up to 20 remote applications.

Message Relay and Locally Known Names

If the application has a locally known name, there are only two ways for a client to communicate with it: by knowing its station name (or PSAP address) or by using broadcast messages. The next section discusses broadcast messages. This one is focussed on those applications that use station names (or PSAP addresses) to find the application.

The INI Message Relay facility can handle programs with locally known names if the locally known program initiates communication. When communication occurs, the Message Relay facility remembers the name and address of the program that sent it a message. This information can be used to return a reply. The locally named program may be on either the Local Station or the Remote Station.

Message Relay and Broadcast Messages

Broadcast message support is required by applications that need to address programs in multiple stations in parallel. In the control network, broadcast messages are used to query all Object Managers about a list of tags and to extract compound lists from Control Stations in support of the Select display.

Most broadcast messages will not be relayed through the INI software. However, some custom applications need this service.

A typical use is the IRIS reporting package, a portion of the RBatch II software offering. IRIS servers may exist on each workstation in the system and IRIS clients need to know which servers are available in the system. The IRIS clients gather this knowledge using broadcast messages. The use of broadcast messages greatly simplifies the configuration of the IRIS package.

The following figure and discussion illustrate the use of broadcast Message Relay.

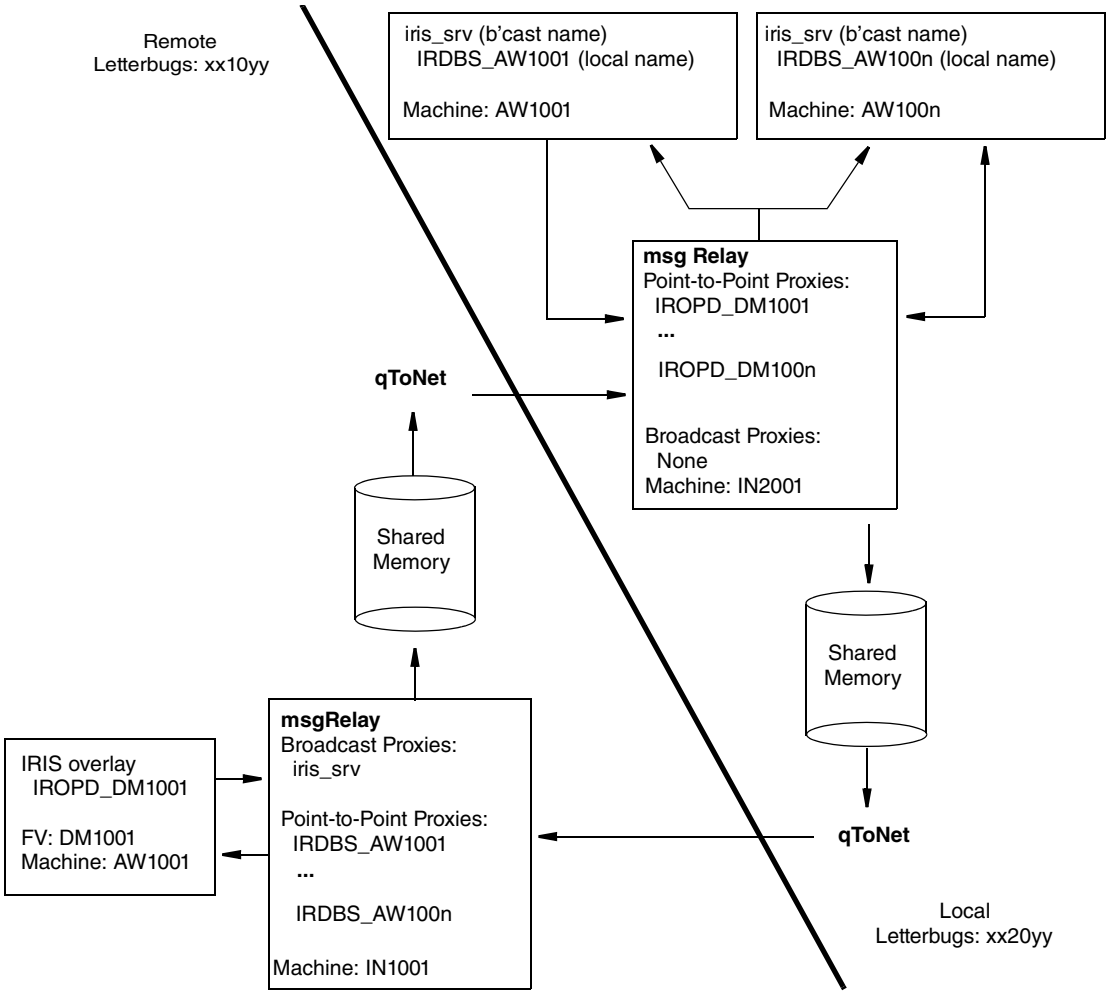


Figure 1-4. Message Relay of Broadcast Messages

The preceding figure provides an overview of the Message Relay process. In this case, the program **msgRelay** on the Local Machine (**IN1001**) is acting as a proxy for a program that requires broadcast Message Relay. The program in this case is a portion of the IRIS component of the Off-sites Automation Suite.

In this example, the IRIS overlay sends a broadcast message to all programs named **iris_srv** on all workstations.

Since the **msgRelay** component of the INI software's Message Relay functionality is acting a broadcast proxy for this name, it receives the message and stores it in the outgoing queue. The

qToNet component takes messages from this queue and forwards them over the CSN to another copy of the **msgRelay** component.

This component, based on information in the message header, places the message on the Control Network using an IPC broadcast message.

The IRIS servers each get the broadcast message and reply to the sending program. They know the local name of the sending program and its address on the network from the information included in the broadcast message.

The **msgRelay** component on the Local Station acts as the proxy for all possible clients on the Remote Station and as their proxy receives the message from the IRIS server and puts it in the outgoing queue. The **qToNet** component takes messages from this queue and forwards them over the CSN to the remote **msgRelay** component.

The remote **msgRelay** component sends the message directly to the IRIS overlay.

NetFoxAPI/NetAIM*API Licensing

The NetFoxAPI/NetAIM*API software license is included as part of the INI deliverable. The NetFoxAPI/NetAIM*API license codes can be updated with a new temporary code or a permanent code without restarting the INI because the NetFoxAPI/NetAIM*API reads the code only on startup.

Refer to “NetFoxAPI/ NetAIM*API Licensing” on page 107 for detailed instructions on obtaining the required NetFoxAPI/NetAIM*API licenses.

2. Assumptions and Limitations

This chapter describes the limitations of the INI product.

Hardware

The INI package requires an Application Workstation as its execution platform on both the Remote Station and Local Station systems.

In addition, the hardware necessary to complete a TCP/IP connection between the two systems must be present and configured. Configuration of such equipment is beyond the scope of this document.

FoxAPI/AIM*API

Both the local and remote workstations must either be using networked FoxAPI or AIM*API. The mixing of networked FoxAPI and AIM*API between the local and remote workstations is not allowed.

Data Transfer Limitations

A TCP/IP network between the local and remote stations must be in place and operating before this software is installed.

The INI package has some limitations:

1. Unlike the INI15, the INI package does not have a dynamic database. Tag names must be configured if they are to be available.
2. Unlike the INI15, INI software tags do not appear in the FoxSelect™ displays. However, if the proper CBPs are monitored, the local detail display file works with the INI tags.
3. *The release of the INI package covered by this document does not directly support redundancy.* However, the non-redundant INI hosts can be attached to a redundant CSN. In this case, the INI package will be immune to cable faults within the CSN, but it will be susceptible to faults in its host, the Network Interface Card (NIC) in the host, and the cable between the CSN and the NIC. In addition, application level redundancy can be provided. See the discussion on page 39.
4. *It is possible for race conditions to exist over an INI software link that are not normally detectable in a Control Network based system.* Some applications use the change of a CBP to signal completion of certain data value changes. In the INI software, it is possible for the change in this flag to be reported before the values that it is signaling are reported. Applications need to be aware of this phenomenon. The same condition could occur in a Control Network based system, but the delays in the INI software make it more pronounced.

5. The INI software handling of Last Good Value (LGV) may differ from expectations. There are two forms of LGV. Restoration of value at bootup and holding values when they go bad.

   The INI software restores the value from the database when an object is created, but it does not update these values except for **U** or **g** maps. There are several reasons for this behavior:

   - While it might be possible to update the database during a normal shutdown, doing so during an abnormal shutdown would be difficult to guarantee.
   - There may be a long time between save and re-create and the value restored might be just as invalid as zero. For this reason, the handling of all such cases are identical.
   - The time it takes to update the database is quite high. If the attempt was made to update it frequently enough to catch a LGV, the load on the machine would increase and the updates would fall behind because the database would not be able to keep up for large applications.

6. If the multi-map mode of the INI executable is used, all of the applications for a given instance must be in the same remote host. That is, a given INI instance can communicate with exactly one remote host. The remote host can, of course, retrieve data from any CP on its Control Network.

   The INI software uses the standard data status flags (BAD and OOS) to indicate that the data value should not be used. Most of the control blocks (PID, RATIO, etc) validate their inputs by checking these bits. At startup, the INI software sets the BAD bit. During loss of communications, we set BAD and OOS. This may mean that control schemes using Sequence Blocks or CALC blocks may need review.

   The second type of LGV is implemented by the INI software. In this case, if a value is reported from the remote site and that value has a BAD bit set or an OOS bit set, we update the local status, but we do not change the local value. The belief is that the value returned with the BAD or OOS bit is unlikely to be useful.

Message Forwarding Limitations

Each instance of the Message Forwarding facility running in the Remote Station is limited to 20 logical names each of which represents 20 targets on the Local Station. The limit of 20 logical names is an IPC limitation.

Multiple copies of the Message Forwarding component may be run to work around this limitation. Each instance will require its own service name.

If the file specifying the alarm message destinations is changed, the Message Forwarding software must be stopped and restarted so that the file is re-read.

The Message Forwarding software is based on the assumption that the remote Compound:Block and the local Application:Object have exactly the same name. The Message Forwarding software does not convert the tag names contained in the forwarded messages.

Message Relay Limitations

Each instance of the Message Relay facility is limited to representing 20 processes in the other system. This limitation is imposed by the IPC mechanism.

In some cases, the limitation can be worked around by running multiple copies of the Message Relay facility in each box. Each instance will require its own service name.

If the file specifying the alarm message destinations is changed, the Message Relay software must be stopped and restarted so that the file is re-read.

3. Installation

This chapter describes the installation of the INI product.

Overview

Installation consists of the following steps:

1. Obtaining the required permanent NetFoxAPI/NetAIM*API software license.
2. Installing any relevant Quick Fixes as per Quick Fix Installation instructions. The FoxAPI/AIM*API software has frequent upgrades so check with the Global Customer Support Center for the latest.
3. Installing the Microsoft SQL Server® software.
4. Install the INI70 software.
5. Configuring the stations Configurable Operating System to support the number of required Application Objects.
6. Configuring the FoxAPI/AIM*API software, a.k.a., AIS.
7. Configuring the NetFoxAPI/NetAIM*API software on both the INI70 host and the machine running the NetFoxAPI/NetAIM*API server supplying data to the INI70 package.

Microsoft SQL Server Software

Microsoft SQL Server® software must be installed before the installation of the INI70 software. The INI for Windows media includes a copy of Microsoft SQL Server 2008 Express Edition (for use with I/A Series software v8.8 and Foxboro Evo Control Core Services v9.0 and higher) and Microsoft SQL Server 2005 Express Edition (for use with I/A Series software v8.2- v8.7).

— NOTE

If Microsoft SQL Server 2008 or Microsoft SQL Server 2005 software is already installed on the workstation, the installation of the Microsoft SQL Server Express Edition software **is not** necessary.

Control Core Services and I/A Series Software v8.8 Instructions

To start the Microsoft SQL Server 2008 Express Edition software installation process:

— TIP

Only use the installation procedures found in this manual to install the Microsoft SQL Server 2008 Express Edition software.

1. For security-enhanced stations, install the Microsoft SQL Server 2008 Express Edition software as a Microsoft domain user that has software installation privileges (i.e. iainstaller).
2. Using the INI for Windows media, start the SQL Server installation process by selecting the `E:\SqlSrvr\SQL2008\SQLEXPR_x86_ENU.exe` file.
3. At the *Open File - Security Warning* window select `Run`.
4. If the *User Account Control* window appears, select `Yes` to allow the installation of the software.
5. At the *SQL Server Installation Center* select `New installation or add features to an existing installation`.
6. At the *License Terms* window accept the license agreement and select `Next`.
7. At the *Features* window take the default settings but change the installation directory to `D:\Program Files (x86)\Microsoft SQL Server\` and select `Next`.
8. At the *Instance Configuration* window change the Instance root directory to `D:\Program Files (x86)\Microsoft SQL Server\` and select `Next`.
9. At the *System Configuration* window change the **SQL Server Database Engine** to `NT AUTHORITY\SYSTEM` and select `Next`.
10. At the *Database Engine Configuration* window take the default settings and select `Next`.
11. At the *Error Reporting* window take the default settings and select `Next`.
12. At the *Complete* window select `Close`.
13. Close the *SQL Server Installation Center* window.

Continue with the installation of the Microsoft SQL Management Studio:

1. At the *Open File - Security Warning* window select `Run`.
2. If the *User Account Control* window appears, select `Yes` to allow the installation of the software.
3. At the *SQL Server Installation Center* select `New installation or add features to an existing installation`.
4. At the *Installation Type* windows take the defaults and select `Next`.
5. At the *License Terms* window accept the license agreement and select `Next`.
6. At the *Features* window take the default settings and select `Next`.
7. At the *Error Reporting* window take the default settings and select `Next`.
8. At the *Complete* window select `Close`.
9. Close the *SQL Server Installation Center* window.

Installation of the Microsoft SQL Server 2008 Express Edition is now complete.

I/A Series Software v8.2-v8.7 Instructions

To start the Microsoft SQL Server 2005 Express Edition software installation process:

— TIP

Only use the installation procedures found in this manual to install the Microsoft SQL Server 2005 Express Edition software.

1. For security-enhanced stations, install the Microsoft SQL Server 2005 Express Edition software as a Microsoft domain user that has software installation privileges (i.e. iainstaller).
2. Using the INI for Windows media, start the installation process by selecting the **E:\SqlSrvr\SQL2005\SQLEXPR.EXE** file.
3. Only use the installation procedures found in this manual to install the SQL Server.
4. At the *Open File - Security Warning* window select **Run**.
5. At the *End User License Agreement* window accept the license agreement and select **Next**.
6. At the *Installing Prerequisites* window select **Install**.
7. When the required components are installed successfully, select **Next**.
8. At the *Welcome to the MSSQL Server Installation Wizard* window select **Next**.
9. At the *System Configuration Check* window select **Next**.
10. At the *Registration Information* window fill-in the site specific information and select **Next**.
11. At the *Feature Selection* window:
    - Select **Browse**
    - Change the installation directory to **D:\Program Files\Microsoft SQL Server**
    - Select **OK**.
    - Select **Next**.
12. At the *Authentication Mode* window select **Next**.
13. At the *Error and Usage Report Settings* window leave both boxes unchecked and select **Next**.
14. At the *Ready to Install* window select **Install**.
15. At the *Completing Microsoft SQL Server 2005 Setup* window select **Finish**.

Continue with the installation of the Microsoft SQL Management Studio:

1. Select the **SQLServer2005_SSMSEE.msi** file.
2. At the *Open File - Security Warning* window select **Run**.
3. At the *Welcome to the Install Wizard for Microsoft SQL Server Management Studio Express* window select **Next**.
4. At the *License Agreement* window accept the license agreement and select **Next**.
5. At the *Registration Information* window select **Next**.

6. At the *Feature Selection* window select **Next**.
7. At the *Ready to Install the Program* window select **Install**.
8. At the *Completing the Microsoft SQL Server Management Studio Express* window select **Finish**.

Installation of the Microsoft SQL Server 2005 Express Edition is now complete.

Stations with Security Enhancements

If the Microsoft SQL Server Express Edition software has been installed on a station with security enhancements, the following modification will need to be made.

1. Select **Microsoft SQL Server 2008 R2** or **Microsoft SQL Server 2005** from the **Start** menu and double-click the **SQL Server Management Studio Express** item.
2. At the *Connect to Server* windows select **Connect**.
3. From the *Microsoft SQL Server Management Studio* select the **Security** folder and then the **Login** folder.
   a. Double-click the **BUILTIN\Users** item from the list.
   b. From the *Login Properties - BUILTIN\Users* window, select the **Server Roles** item and place a check in the **public** box and select **OK**.
4. Close the *Microsoft SQL Server Management Studio* window.

INI70 Software Installation

This section describes the installation of the INI70 software on a system.

— TIP —

Only use the installation procedures found in this manual to install the INI70 software.

1. For security-enhanced stations, install the INI70 software as a Microsoft domain user that has software installation privileges (i.e. iainstaller).
2. Verify the FoxAPI/AIM*API software is installed and configured.
3. Install any relevant Quick Fixes per the Quick Fix installation instructions. The FoxAPI/AIM*API software has frequent upgrades, so check with the Global Customer Support Center for the latest information.
4. Verify that the Network Interface Card (NIC) has the power saver feature disabled.
   a. Open the Microsoft Control Panel.
   b. Select **Network Connections**.
   c. Right-click the NIC to be used for the INI communications and select **Properties**. For example: Local Area Connect (Broadcom NetXtreme 57xx Gigabit Controller).
   d. From the Local Area Connection Properties window select **Configure**.
   e. From the NIC windows select the **Power Management** tab.

f. **Uncheck** the "Allow the computer to turn off this device to save power" box and then select **OK**.

5. Insert the INI for Windows media into the CD-ROM drive in the workstation.
6. Start the installation process by selecting **My Computer**, then the CD-ROM drive letter (for example **E:**).
7. Select the **E:\INI70** folder and then select the **setup.exe** file.
8. After the installation is complete, remove the installation media and reboot the workstation.

4. Configuration

This chapter describes how to configure the INI software.

Licensing

The INI software is a licensed product. When the INI product is ordered, a certain number of software licenses are purchased.

This number controls two aspects of the INI package:

- The number of (local) stations that may host the INI software
- The number of (remote) stations that may supply data to the INI software

There are two types of INI licenses available:

- The INI:Asymmetric
- The INI:Symmetric

The asymmetric license allows the INI software to be installed on a workstation such that it can obtain information from a remote workstation. This allows the local system to have access to remote data and messages, but the remote system does not have access to local resources.

The symmetric license allows the INI software to be installed on each end of the link such that each end has access to the other end's resources.

Multiple instances of the INI executable may be run on a given workstation under a single license.

INI: Asymmetric

The INI Asymmetric license is the simplest INI configuration. The INI software is loaded on a local workstation and the NetFoxAPI/NetAIM*API Server is licensed for one client on the remote workstation. The following figure illustrates the situation.

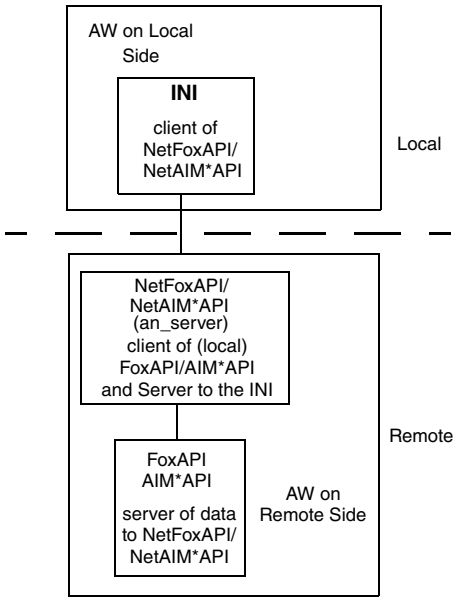


Figure 4-1. Asymmetric Licensed Configuration

In this configuration, the local workstation represents the local data in the remote station.

INI:Symmetric

The INI Symmetric license is the equivalent of loading INI:Asymmetric licenses on each workstation at the either end of the CSN. INI:Symmetric licenses are sold such that the licensed number of stations can communicate with the each other in a fully symmetric manner. The following figures illustrate common combinations of equipment.

Single INI:Symmetric License

With a single INI:Symmetric license, the user may install the INI software on two workstations and configure them to exchange data as shown in the following diagram:

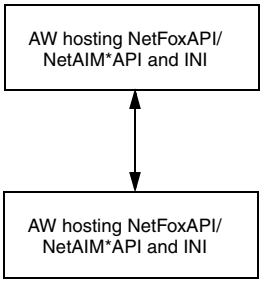


Figure 4-2. Single INI Symmetric License

In this configuration, each workstation is running exactly one copy of the INI and that one copy is talking to a NetFoxAPI/NetAIM*API server licensed for one client.

Five INI:Symmetric Licenses

A set of INI:Symmetric licenses allows more flexibility in the supported configuration.

The default configuration is simply five pairs of workstations exchanging data as shown below.

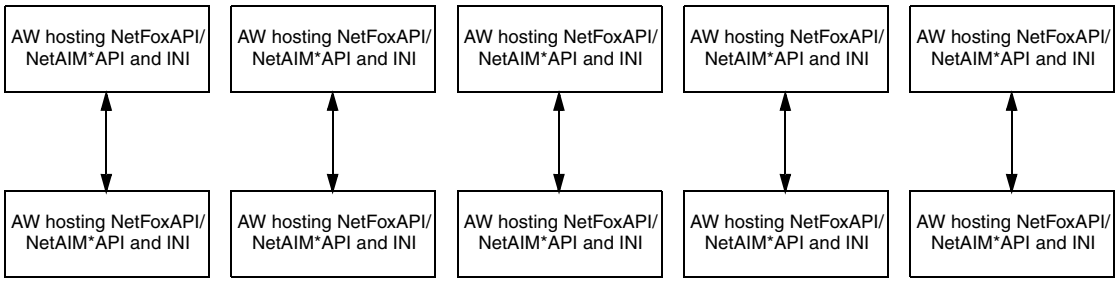


Figure 4-3. Default Five INI: Symmetric Configuration

Each of these pairs is identical to the single license case. The difference is in the pricing of the software.

However, the license may also be used to configured to support full exchange between five machines as follows:

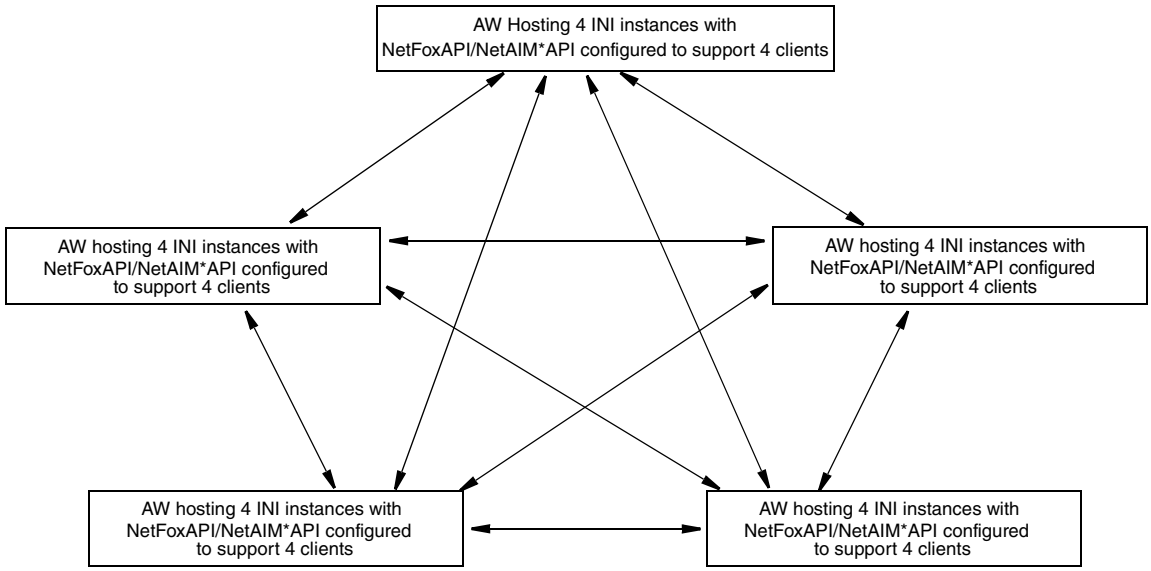


Figure 4-4. Five INI: Symmetric Licenses used for full Exchange of Data

In the above diagram, each workstation is running four copies of the INI software executive and each workstation is running one copy of the NetFoxAPI/NetAIM*API software configured to support four users.

NetFoxAPI/NetAIM*API Licenses

The NetFoxAPI/NetAIM*API software license is included as part of the INI deliverable. The NetFoxAPI/NetAIM*API license codes can be updated with a new temporary code or a permanent code without restarting the INI because the NetFoxAPI/NetAIM*API reads the code only on startup.

Temporary licenses are granted freely since they expire after a period of time. It is recommended that temporary licenses be used until the final workstations are selected. Re-issuing permanent licenses is difficult.

A temporary license will not expire while the software is in use. If the expiration date passes, the software will continue to function until it is stopped. The software will not restart if the expiration date has passed.

Refer to “NetFoxAPI/ NetAIM*API Licensing” on page 107 for detailed instructions on obtaining the required NetFoxAPI/NetAIM*API licenses.

See *I/A Series System FoxAPI Installation Guide* (B0193UC) for instructions in the use of **an_setup** to enter license codes.

See *AIM*AT Suite AIM*API User’s Guide* (B0193YN) for instructions in the use of **an_setup** to enter license codes.

Configuring the System Parameters

By default, stations have the ability to create 1250 Shared Variables. Because Application Objects use the same resources as Shared Variables and because some user applications require several hundred Application Objects each with a hundred or so attributes, one must raise this limit.

The normal maximum limit is 4000. This can be increased to as much as 40,000. This secondary limit may be raised in turn by altering other files.

However, the performance of the workstation can be affected by this setting if the workstation does not have enough RAM to handle this and other demands placed on it.

While it is the user's responsibility to purchase enough RAM for the required applications, the INI package includes a tool to help size its requirements.

Calculating the Correct Setting for OM_NUM_OBJECTS

— NOTE

For more information on OM_NUM_OBJECTS and the OS configurable parameters that enable you to fine tune OS extension resources for a particular application, refer to *The MESH Control Network System Planning and Sizing* (B0700AX).

The ratio of AOAs to OM Shared Variables (the value of **OM_NUM_OBJECTS**) is approximately 2.5:1.

The ratio of RAM (bytes) to **OM_NUM_OBJECTS** is approximately 350 bytes per OM variable.

However, the better technique to use in the calculation of **OM_NUM_OBJECTS** setting is to use the INI tool **calcAppSize** on the map files. See page 74 for instructions on how to use **calcAppSize**.

— TIP

Do not use the calcAppSize tool until a map file has been created.

Note that **calcAppSize** program calculates only the space required to hold the Application Object Attributes. Additional space is required to hold the OM lists that may be generated by applications that use these values. Each OM list requires 64 bytes for a header and 64 bytes for each variable. Using this information plus the fact that each **OM_NUM_OBJECTS** increment make 350 bytes available, one can estimate the additional increase required to support these lists.

For example, if one has 20 display managers each looking at 75 AOAs, the increase in the setting of **OM_NUM_OBJECTS** is approximately:

$$\frac{) * (64 * (1 + 75))}{350} = 27$$

Generally, it is easier to simply round the output of **calcAppSize** up a few hundred.

Altering the Sizing Files

The procedure to increase the number of OM objects in a workstation is:

1. Run `D:/usr/fox/exten/config` and record the results.
2. Edit the file `D:\usr\fox\exten\config\user_rules.cfg` to change the `OM_NUM_OBJECTS` variable to the computed value, if it is larger than the default value.
3. Edit `D:/etc/fox/opsys_usr.cfg` to change the tope range on `OM_NUM_OBJECTS` to the computed value as described above if the calculated value is larger than the current value.
4. Run `d:/usr/local/reconfig_IA prep_reboot` to change the parameters on the next reboot.
5. Reboot the workstation.
6. Monitor memory paging to ensure that the machine has enough RAM to minimize the performance impact.

Configuring the I/O Subsystem

NetFoxAPI/NetAIM*API and FoxAPI/AIM*API

The system has the following APIs available to obtain process data:

- The Object Manager (OM) API,
- The local FoxAPI/AIM*API, and
- The NetFoxAPI/NetAIM*API.

The Object Manager is the fundamental API for accessing process data in the system. However, this API was not designed for the casual or inexperienced user. For this reason, FoxAPI/AIM*API software was developed.

Though it uses the OM API, the local FoxAPI/AIM*API application hides a significant amount of the OM API's complexity from the user.

The NetFoxAPI/NetAIM*API software was developed to extend FoxAPI/AIM*API applications over a variety of networks to various client computers and Operating Systems. In addition, it further simplifies the use of the OM and minimizes the consumption of various resources used by the OM.

The INI software requires the latest FoxAPI/AIM*API version on the remote side of the connection and the corresponding client software on the INI side of the connection.

The FoxAPI configuration files are:

- **foxapi.cfg** - A Remote Station File
- **an_init.tcp** - A Remote Station File
- **an_init.cfg** - A Local Station File

The AIM*API configuration files are:

- **aimapi.cfg** - A Remote Station File
- **an_init.tcp** - A Remote Station File
- **an_init.cfg** - A Local Station File

*NetFoxAPI/NetAIM*API Configuration Files*

The NetFoxAPI/NetAIM*API configuration files to be edited are **an_init.tcp** and **an_init.cfg**. Use any text editor to edit the files and saved them as a plain text file.

The NetAIM*API configuration files to be edited are **an_init.tcp** and **an_init.cfg**. Use any text editor to edit the files and saved them as a plain text file.

Likewise, the local FoxAPI/AIM*API configuration file should be edited in the same manner. The following sections explain the contents of these files.

*FoxAPI/AIM*API Configuration Files*

foxapi.cfg/aimapi.cfg - A Remote Station File

The **foxapi.cfg/aimapi.cfg** file on the Remote Station must be modified to ensure that FoxAPI/AIM*API application has adequate resources to support the INI software. The document *FoxAPI Information Bulletin* included in the INI deliverables contains information on the settings found in this file. Several modifications are common:

- Modify the maximum number of objects that FoxAPI/AIM*API application may address (**maxobj**) from the default of 5000 to a more useful number, such as 10000 or 20000.
- Modify the maximum number of data sets (**maxds**) that FoxAPI/AIM*API clients may use from the default of 20 to at least 40 or more if many Applications with **W** mappings are to be run. The INI uses a data set for **W** mappings. If there are no **W** mappings, the limit does not need to be changed.
- If large numbers of points are being opened in a short period of time, ensure that **ctdlay** is set to 200 (2 seconds).

See page 103 for an example of the **foxapi.cfg/aimapi.cfg** file.

an_init.tcp - A Remote Station File

The **an_init.tcp** file contains the NetFoxAPI/NetAIM*API authorization code. See page 104 for an example of the an_init.tcp file.

The NetFoxAPI/NetAIM*API authorization code is associated with the variable **SUNSPARC_AISnet** found in the file. See page 107 for detailed instructions on obtaining the NetFoxAPI/NetAIM*API license codes.

It is strongly recommended that temporary license codes be used until one is certain which workstation will be used to host the NetFoxAPI/NetAIM*API Servers that feed data to the INI software.

The NetFoxAPI/NetAIM*API program allows the authorization code to be changed at any time. The code is only checked when NetFoxAPI/NetAIM*API application is started and an expired temporary code will not cause the INI software to cease operation. Use a text editor to create or revise this file in the **D:\opt\fox\ais\bin** directory.

an_init.cfg - A Local Station File

The **an_init.cfg** file is used by the INI software to determine the NetFoxAPI/NetAIM*API Server that should used. See page 105 for an example for the **an_init.cfg** file.

— NOTE

The most important entry in this file is the **Host** line. This line specifies the name of the remote station. The remote station name must be listed in the [**TCPIP**] section of the **an_init.cfg** file.

If the INI executable is being run directly, this file should be in the directory from which the INI is launched.

If the INI mapping scripts (**INIMAP** and **INIMMAP**) are used to start the INI executable, the INI executable will be started in the directory **D:\opt\aos\data** unless the **-M** option is specified. If the **-M** option is used, the INI executable is started in the directory specified by the **-M** option. Because the script changes the directory from which the INI is started, the INI executable will not look for the an_init.cfg file in the directory where the script was started. Rather it will look in either the **D:\opt\aos\data** directory or the directory specified by the **-M** option since one of those will be its current directory.

Regardless of how the INI executable is started, the **-AN** option to the script overrides this behavior and directly specify the location of the an_init.cfg file.

The an_init.cfg file must be created on the Local Station using a text editor.

Follow the procedures in the *I/A Series System FoxAPI Installation Guide*, B0193UC to create an an_init.cfg file for your system.

Error Logging

The INI70 package uses the Windows Event Viewer as the log of last resort. This service is always enabled. Normal maintenance procedures apply.

Data Transfer

Overview

The following diagram shows the work that must be performed on the Remote Station to create the INI's input (map) file:

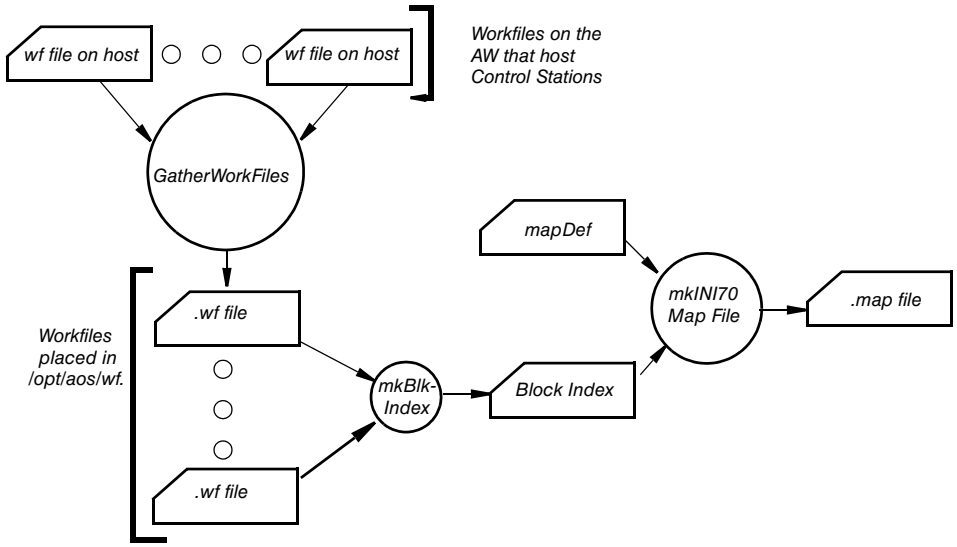


Figure 4-5. Remote Station: Creating map Files from Control Station Workfiles

Once the map files have been created on the Remote Station, copy them to the project directory on the Local Station. See page 43, for more information regarding the project directory.

The following diagram shows how that map files are turned into an Application once the map files are copied to the project directory on the Local Station.

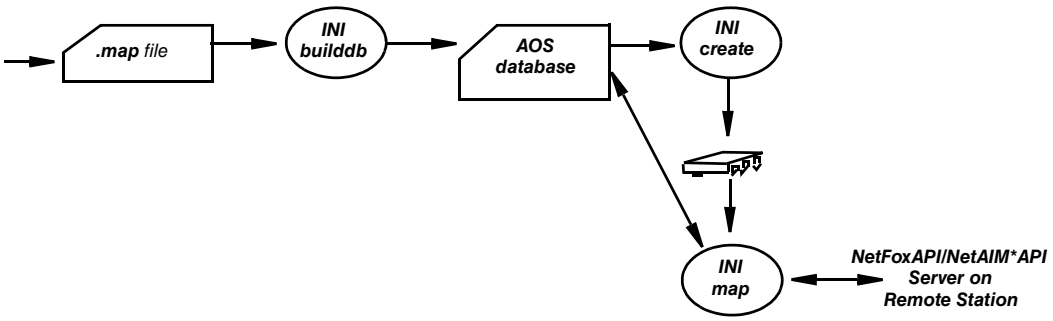


Figure 4-6. Local Station: Creating an Application from a map file

The information supplied by the user of the INI package is a collection of work files and a Map Definition file. The work files contain the information about the remote stations and can be obtained in a variety of ways. The most common of which is the tool **GatherWorkFiles**, see page 61.

The gathered work files are indexed using the tool **mkBlkIndex**, see page 83. The index produced by this program and the user supplied map definition (**D:\opt\aos\data\mapDef**) file are

combined by the tool **mkINIMapFile**, see page 84, to create a map file for each compound of interest.

The map file defines all of the objects belonging to an Application, their attributes, data types, default values, and change deltas. This file is read in by the INI software and its information is stored in an database.

The database is queried by the INI software to create the application objects used to store the remote data. These objects are stored in the memory of Local Station.

The INI executable is used to convert the map file, to create the application objects, and to map the data from the Remote Station to the Local Station and vice versa. The script **go_INI70** can be used to re-create the objects and start the transfer of data at reboot. See page 70 for information on **go_INI70**.

Map File Generation

A map file (page 90) is required to define the operation of the INI package. In addition, the INI package includes three tools to facilitate the creation of a map file.

These tools are:

- **GatherWorkFiles** - a script that gathers all workfiles into one directory, see page 61.
- **mkBlkIndex** - a program that reads the workfiles and generates an index of all blocks in the workfiles, see page 83.
- **mkINIMapFile** - a program that creates map files that contain records for specified parameters of specified block types, see page 84.
- **mapDef** - A plain text file that specifies which parameters are to be extracted from the workfiles, see page 90.

Tool and file specific documentation can be found on the indicated pages. Use the information on those pages to alter the file to meet the needs of the project.

The general procedure for the use of these tools is:

- Load the tools on the Remote Station by loading the INI package on the Remote Station.
- Change to the INI package's scripts directory: `cd D:/opt/aos/scripts`
- Clear the work directories of any old files:
  `rm -r D:/opt/aos/wf/* D:/opt/aos/tmp/*`
- Collect all of the work files into one location (**D:/opt/aos/wf**):
  `./GatherWorkFiles`
- Build a block index for the collected workfiles:
  `../exe/mkBlkIndex ../wf/*/*.wf | sort -o ../wf/BlockIndex`
- Revise the **BlockIndex** file to remove blocks of no interest.
- Revise **mapDef** to specify the block types and parameters of interest.
- Build the map files using a set of project specific rules:
  `../exe/mkINIMapFile ../wf/BlockIndex -d ../tmp <../data/mapDef`
- Copy generated map files (`d:\opt\aos\wf\*`) to the Local Station.
  `d:\opt\aos\wf\*`

Database Initialization

Once the map files have been created and transferred, they need to be loaded into the database by running the script **INIBUILDDB**, see page 62.

Manual Startup

This section discusses how to setup and start a multiple application instance of the INI executable. The names of the applications to be mapped are: **APP_01**, **APP_02**, and **APP_03**. The directory **D:\opt\foxind\INIApps** is the project directory. See page 43 for a suggested layout for your projects.

With the files in their standard locations, the steps to use multimapping are:

1. Populate the database with the definition of the objects using **INIBUILDDB**. The directory path to the map file has to be specified with this command.
2. Create the objects using **INICREATE**. This command does not require knowledge of the map files. It reads the database.
3. Start mapping of multiple applications using **INIMMAP**.

Here are the actual steps for three applications: **APP_01**, **APP_02**, and **APP_03**.

```
cd D:/opt/foxind/INIApps
INIBUILDDB APP_01/data APP_01 APP_01.map
INIBUILDDB APP_02/data APP_01 APP_02.map
INIBUILDDB APP_03/data APP_01 APP_03.map
INICREATE APP_01
INICREATE APP_02
INICREATE APP_03
INIMMAP APP_01 APP_02 APP_3 -SR 5 /D APP_01/logs
```

The logs will be placed in **D:\opt\foxind\INIApps\APP_01\logs**

If the environment is setup properly, the INI executable may be used directly:

```
cd d:/opt/foxind/INIApps
INI70 builddb APP_01 APP_01/data/APP_01.map
INI70 builddb APP_01 APP_02/data/APP_02.map
INI70 builddb APP_01 APP_03/data/APP_03.map
INI70 create APP_01
INI70 create APP_02
INI70 create APP_03
INI70 mmap APP_01 APP_02 APP_3 -SR 5 /D APP_01/logs
```

Once the applications have been built using either **INIBUILDDB** or '**INI70 builddb...**', they do not need to be rebuilt unless the mix of objects and attributes needs to be changed.

Once the objects have been created using **INICREATE** or '**INI70 create...**', they do not need to be recreated unless they have been deleted or the station has been rebooted.

Automatic Startup

At reboot, there are two operations that must occur for each application assuming that the application definitions have been stored previously in the database using **INIBUILDDB** or **INI70 build**.

- Creation of the Application Objects and
- Startup of the mapping service.

The following steps must be performed to automatically start on reboot.

1. Copy the `d:/opt/aos/scripts/go_INI70.ksh` file to the `d:/usr/fox/bin` directory.
2. Edit the `go_INI70.ksh` script to start the specific INI application on reboot.

   ```
   d:/nutc/mksnt/wstart -bw d:/nutc/mksnt/sh
   /opt/aos/scripts/go_INI70
      <ProjectDir> <AppName>
   ```

   where:

   <ProjectDir> is the name of the INI application directory.

   <AppName> is the name of the INI application to be started.
3. Add the following line to the end of the `d:/usr/fox/bin/fox_apps.dat` file.

   ```
   INI70
   ```

Message Forwarding Configuration

Local Side Configuration

Define the Service

In order to receive communication from the Remote Station, the Message Forwarding component that runs on the Local Station must be defined in the file **c:/windows/system32/drivers/etc/services**.

Search the file services for any available <port> numbers greater than 5000, e.g. 15000. The port number is specified in the second token on each line and precedes the slash.

Once a free port number has been identified, insert the following line:

```
IAMsgFwd      <port>/tcp     #I/A Series Message Forwarding support
```

where:

<port> is a number greater than 5000, e.g., 15000, and that is not already in use on this Local Station OR the associated Remote Station.

After the file has been changed, execute the following command to cause the system to re-read the **c:/windows/system32/drivers/etc/services** file:

```
D:/opt/aos/MsgFwd/scripts/tellInetd
```

Define the Message Destinations

The ultimate destination of messages routed to another Local Station must be defined in the file named **D:\opt\aos\MsgFwd\data\FileOfNames**. Use any text editor to customize the file **D:\opt\aos\MsgFwd\data\FileOfNames**. Other file names and locations may be used, see the write-up on page 78.

The format of the file is given on page 87.

Please note that this file exists on both the Remote Station and Local Stations. Any time the file is modified, the updated version must be distributed to both locations.

— NOTE

If modifications have been made to this file after the software has been started, the Message Forwarding software must be stopped and restarted so that the file is re-read.

Start the Software on Bootup

The netToIA component of Message Forwarding must be activated at bootup. For automatic start up, the **go_INI70.ksh** script must be exist in the **D:/usr/fox/bin** directory. Refer to page 33 for automatic startup instructions.

The following line can be added to an existing **go_INI70.ksh** file.

```
d:/nutc/mksnt/wstart -bw d:/opt/aos/MsgFwd/exe/netToIA
```

Be sure to read page 80 and modify the arguments to **netToIA** suit your particular case.

Configuration Check

There are two techniques for checking the configuration. One requires rebooting the station and the other does not. The reboot option is preferred since it is what will happen in the normal operation. However, it is recognized that rebooting is not always an option.

If rebooting the station is an option, reboot the station. When the FoxView display appears on the workstation run the following command from the command prompt:

```
D:/opt/aos/MsgFwd/scripts/MsgStatus
```

The following result should be printed on your screen within a few seconds:

```
iaToQ is NOT alive!
qToNet is NOT alive!
netToIA is alive!
```

If rebooting the station is not an option at installation time, type the following commands:

```
D:/opt/aos/MsgFwd/exe/netToIA
D:/opt/aos/MsgFwd/scripts/MsgStatus
```

The following result should be printed on your screen within a few seconds:

```
iaToQ is NOT alive!
qToNet is NOT alive!
netToIA is alive!
```

In either case, the process **netToIA** must be the only process that is alive on the local station. If this is not the result, then something is wrong. Please review the installation steps for accuracy.

Remote Station Configuration

Update the Configuration File

Once the remote alarm destinations have been defined in the file **FileOfNames** on the Local Station, copy it to the Remote Station.

Modify Alarm Destinations

Once the remote alarm destinations have been defined in the file **FileOfNames**, use a configurator to modify the alarm destination for each affected Block so that the alarm messages are sent to the appropriate destination. The parameters of interest are the Compound parameters **GR1DV1-8**, **GR2DV1-8**, **GR3DV1-8** and Station Block parameters **DV1-16** and **GR4-8.**

Define the Services

In order to receive a message from the Remote Station, the Message Forwarding component that runs on the Local Station must be included in the **c:\windows\system32\drivers\etc\services** file. See the section on Service Definition for the Local Station on page xvi.

— NOTE

The port must be the same number (e.g., 15000) on both the Remote Station and Local Stations.

Start the Software on Bootup

The components of the Message Forwarding component of the INI package that reside on Remote Station must be activated at bootup. For automatic start up a **go_INI70.ksh** script must be exist in the **d:\usr\fox\bin** directory. Refer to page 33 for automatic startup instructions.

The following line can be added to an existing **go_INI70.ksh** file.

```
d:/nutc/mksnt/wstart -bw d:/opt/aos/MsgFwd/exe/iaToQ <serviceName>
<serverName>
```

where

<serviceName> is the name of the entry placed in the **services** file during configuration. See page 33 for details.

<serverName> is the 2nd Ethernet port name of the Local Station.

See page 78 for the details of the **iaToQ** command.

Configuration Check

To ensure that everything has been installed correctly, reboot the Remote Station. When the FoxView display appears, use any text editor and type the following:

```
d:/opt/aos/MsgFwd/scripts/MsgStatus
```

Within a few seconds the following results should display: `i`

```
iaToQ is alive!
qToNet is alive!
netToIA is NOT alive!
```

If this is not the result that you get, something is wrong. Please review the installation steps to make sure you have not forgotten something.

If rebooting the Remote Station is not an option at installation time, perform the following step instead:

```
d:/opt/aos/MsgFwd/exe/iaToQ <serviceName> <serverName>
```

Remember to replace **<serviceName>** and **<serverName>** with the proper names for this installation. Wait several minutes and then verify that the software is running by using the **MsgStatus** script mentioned in the previous step.

Message Relay Configuration

Local Side Configuration

Define the Services

In order to receive communication from the Remote Station, the Message Relay component that runs on the Local Station must be defined in the **c:\windows\system32\drivers\etc\services** file.

Search the file **c:\windows\system32\drivers\etc\services** file for any available **<port>** numbers greater than 5000, e.g. 15000. The port number is specified in the second token on each line and precedes the slash.

Once a free port number has been identified, insert the following line:

```
IAMsgRelay      <port>/tcp      #I/A Series Message Relay support
```

where:

<port> is a number greater than 5000, e.g. 15001 and that is not already in use on this Local Station OR the associated Remote Station.

After the file has been changed, execute the following command to cause the system to re-read the **c:\windows\system32\drivers\etc\services** file.

```
d:/opt/aos/MsgRelay/scripts/tellInetd
```

Define the Message Destinations

There are two files that must be configured and they are complements of each other. The format of these files is given on page 89. The files are:

- **FileOfProxies** - This file resides on the Local Station and contains the names of the processes on the Remote Station with which communication is required. These programs can be considered servers.
- **FileOfTargets** - This file resides on the Remote Station and contains the names of the processes on the Local Station with which communication is required. These programs can be considered clients.

By default, these files are assumed to reside in **d:\opt\aos\MsgRelay\data**. However, they may be kept in another location to simplify maintenance of the system. See page 73.

— NOTE

If modifications have been made to this file after the software has been started, the Message Relay software must be stopped and restarted so that the file is re-read.

Start the Software on Bootup

For automatic start up a **go_INI70.ksh** script must exist in the **D:\usr\fox\bin\fox_apps.dat**. Refer to page 33 for automatic startup instructions.

The following line can be added to an existing **go_INI70.ksh** file.

```
d:/nutc/mksnt/wstart -bw d:/opt/aos/MsgRelay/exe/msgRelay <msgRelayName>
<rmtStationName> -p <fileOfProxies>
```

See page 73 for details regarding the **msgRelayName** command.

Configuration Check

There are two techniques for checking the configuration. One requires rebooting the station and the other does not. The reboot option is preferred since it exactly the normal procedure. However, it is recognized that rebooting is not always an option.

If rebooting the station is an option, reboot the station. When the FoxView display appears on the workstation run the following command from the command prompt:

```
ps -ef | grep -v grep | grep msgRelay
```

The following result should be printed on your screen within a few seconds and it should resemble:

```
1  3660  0  0  17:34:51 CONIN$  0.00  MSGRELAY
```

If rebooting the station is not an option at installation time, type the following command:

```
d:/opt/aos/MsgRelay/exe/msgRelay <msgRelayName> <rmtStationName> -p
<fileOfProxies>
ps -ef | grep -v grep | grep msgRelay
```

Something similar to the following should be displayed:

```
1  3660  0  0  17:34:51 CONIN$  0.00  MSGRELAY
```

Remote Station Configuration

Update the Configuration File

Once the **FileOfProxies** and the **FileOfTargets** files have been created on the Local Station, the **FileOfTargets** must be moved to the Remote Station.

Define the Services

In order to receive a message from the Remote Station, the Message Relay component that runs on the Local Station must be included in the **c:\windows\system32\drivers\etc\services** file. See the section on Service Definition for the Local Station on page 33.

— NOTE

The port must be the same number (e.g., 15001) on both the Remote Station and Local Stations.

Start the Software on Bootup

For automatic start up a **go_INI70.ksh** script exist in the **d:\usr\fox\bin\fox_apps.dat** directory. Refer to page 33 for automatic startup instructions.

The following line can be added to an existing **go_INI70.ksh** file.

```
d:/opt/aos/MsgRelay/exe/msgRelay <msgRelayName> <localStnName> -p <fileOfTargets>
```

See page 82 for details regarding the **msgRelay** command.

Configuration Check

To ensure that everything has been installed correctly, reboot the Remote Station. When the FoxView display appears, open a command window and type the following:

```
ps -ef | grep -v grep | grep msgRelay
```

The following result should be printed on your screen within a few seconds. The output should resemble:

```
1  3660  0  0  17:34:51 CONIN$  0.00  MSGRELAY
```

5. Additional Configurations

This chapter discusses some optional INI configurations that can be made after the installation of the INI software.

Data Redundancy

Redundant analog inputs are handled as follows:

1. Setup INI instances on two different machines.
2. Use the same map file on each machine, but specify a different application name on the backup machine, e.g., **TOWER1_P** and **TOWER1_S**. The map file does not need to change since the application name in it is overridden by the arguments to the INI **builddb**, **create**, and **map** commands.
3. In a FT CP, build a **SIGSEL** block for each value that needs to be redundant with the following settings:
   - COMPOUND based on name of compound in remote system, e.g., **TOWER1**
   - BLOCK set to name of block in remote system, e.g., **FT101** or **FT101SSEL** (if using AIN, see below)
   - NUMIMP set to 2
   - PROPT set to 1
   - EROPT set to 2
   - SELOPT set to 1 (high select)

   This configuration causes "lost" inputs to be skipped automatically based on their status bits. With a **SWCH** block you would have to have logic to determine if a failure occurred and toggle the switch. This configuration is simpler.
4. If alarms are desired or the parameter name needs to be .PNT then, in the same CP, build an AIN block configured as follows:
   - COMPOUND set to name of compound in remote system, e.g., **TOWER**
   - BLOCK set to name of block in remote system, e.g., **FT101**
   - IOMOPT set to 2
   - MEAS set to C:B.OUT of the corresponding SIGSEL block, e.g., **TOWER:FT101SSEL.OUT**

Redundant analog outputs are handled by using the INI in symmetric mode and following the instructions for analog inputs on the remote machine.

Redundant contact inputs are handled by the same mechanism as for analog inputs with the exception that the output of the SIGSEL is run through a CIN to get it back to a Boolean data type.

Redundant contact outputs are handled on the remote system as contact inputs using the INI in symmetric mode.

Message Redundancy

The message receiving software can be run on multiple workstations on each end. Simply ensure that the message is sent to both instances of **iaToQ**.

INI15 Replacement

The INI15 and the INI provide similar functionality. However, they use different equipment and offer different services.

The major differences are shown in the following table:

Table 5-1. Comparison as INI15 and INI

Feature	INI15 Implementation	INI Implementation
Physical Layer	Serial	Ethernet
Protocol	INI15 protocol over X.25	NetFoxAPI/NetAIM*API over TCP/IP
Remote data access	Dynamically accessible Requires 2-character prefix on remote compound name. Configured by entering compound names and prefix into the INI15 configuration.	Static configuration with ability to add CBPs on-line. Uses same name on both systems by default. The **mkINIMapFile** tool understands the INI15s naming convention and can preserve that name. Configured by building a Map File. Generally, the tools **GatherWorkFiles**, **mkBlkIndex**, and **mkINIMapFile** are used.
Process Alarms, SOE messages, and Sequence Block messages	Requires optional software running on Venix based host, i.e., AP20 or PW	Requires optional software running on the same machine.
Other IPC messages	No support.	Supported.

In general, INI15s operate over dedicated telephone line. Through the use of equipment like CSU/DSU units and routers, these lines can typically be reused by the INI.

After the TCP/IP network between the two systems is established, the major issue is the creation of the map file. The normal tools support the INI15 name configuration. This support allows the reuse of all of their engineering work. See page 84 for a description of **mkINIMapFile**.

Splitting a Node

If the INI package is to be used to re-link a split Node or to re-line a Node moved off of a CBLAN, the compound names moved to the "remote" system must be removed from the Local Station and vice versa. Standard tools are available to remove those names.

The CSA tool **csa_fn** should be used to remove the compound names from the stations that no longer exist in the split CBLANs.

For example, if after the split, **Node A** no longer has the station **CP00A1** because that station now resides in **Node B** then **csa_fn** should be used to eliminate from **Node A**'s CSA the compound names that belong to **CP00A1**.

This is done by typing:

```
D:/usr/fox/csa csa_fn reset CP00A1
```

This will release the "moved" compound names so that the INI can use them.

This command should be repeated in **Node A** for each station that is now in **Node B**. Similarly, compounds in stations in **Node A** must be eliminated from **Node B**.

6. Operation

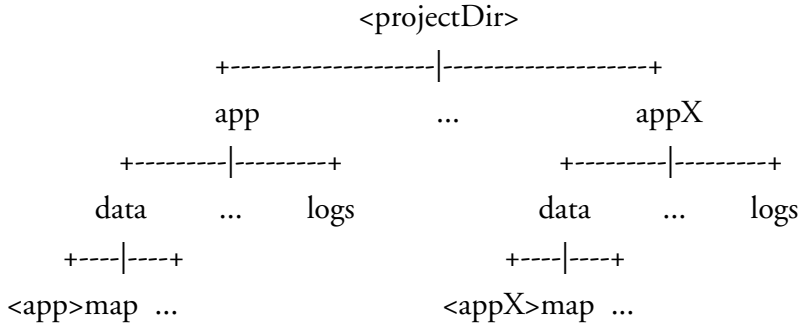
The operation of the INI software is described in this chapter.

Data Transfer

The data transfer component moves remote data into local AOAs. The next several sections explain how to best use this service.

Project Directory

If applications are going to be mapped individually, the INI software works best if the files required to run it are organized as follows:



where:

<projectDir> is the directory that holds all the INI files.

<app> ... <appX> are directories that hold files for a specific application.

<projectDir>\app>\<data> holds the **<app>.map** file.

<projectDir>\<app>\<logs> holds the log files.

If the multiple map capability of the INI software is going to be used so that a single instance of the INI process is mapping multiple applications, it may be easier to place all of the map files in a single directory.

In this case, the **go_INI70** script will create the individual directories for each application and use those log files for the **aos_create** logs. The mapping logs will go in the logs directory of the first application named on the startup line.

General

The INI package includes tools for performing most common operations. These operations are:

- Map File Generation - Extracts needed information from the Remote Station's Control Station workfiles.
- Build - Converts the map file to records in the database.
- Create - Reads the database and creates the Application Objects in the RAM of the Station.
- Map - Starts data transfer.
- Monitor - Monitors the operation of an INI instance.
- Checkpoint - Checkpoints values in the Application to the database.
- Re-Read - Re-reads the **G/g** mappings.
- Unmap - Stops data transfer.
- Delete - Removes the Application Objects from memory.

The normal development sequence is:

- Build a map file.
- Build the application in database (**INIBUILDDB**).
- Create the objects (**INICREATE**).
- Start mapping (**INIMAP** or **INIMMAP**).
- Stop mapping (**INIUNMAP**).
- Delete the objects (**INIDELETE**).

At startup/reboot, the sequence is:

- Create the objects (**INICREATE**).
- Start mapping (**INIMAP** or **INIMMAP**).

Generally, the startup steps are handled by the **go_INI70** script.

There is no need to re-build the database at reboot.

Status Monitoring

To use the INI Monitoring package, the following must be done:

- The INI Application definition directories must be created as described on page 43.
- The directory **D:\opt\aos\Monitor\disp** must be attached to a menu button in an environment.
- The Applications must be loaded initially into the database using the script **INIBUILDDB** or the **INI70** command directly.

Once the applications have been initially loaded, the script **mkMonitorList** in the **D:\opt\aos\monitor\scripts** directory may be run to create the files necessary to support the Monitor package.

Once the **mkMonitorList** script has been run, the AOSMon menu picks for **D:\opt\aos\Monitor\disp** will resemble the display in Figure 6-1.

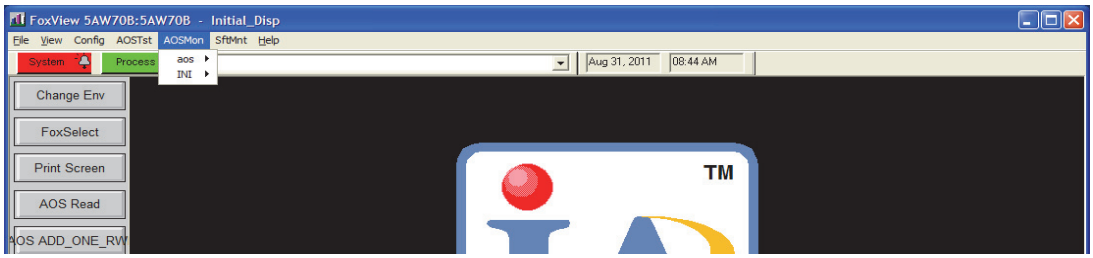


Figure 6-1. Monitor Package Main Menu

From the AOSMon menu, the INI based applications can be monitored by selecting the `INI` menu item, Figure 6-2.

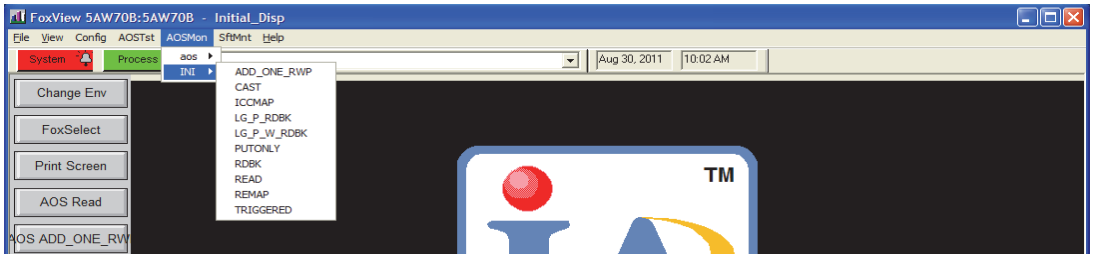


Figure 6-2. Monitor Package with List of Applications to Monitor

Once an application is selected, its respective status monitoring display (Figure 6-3) appears.

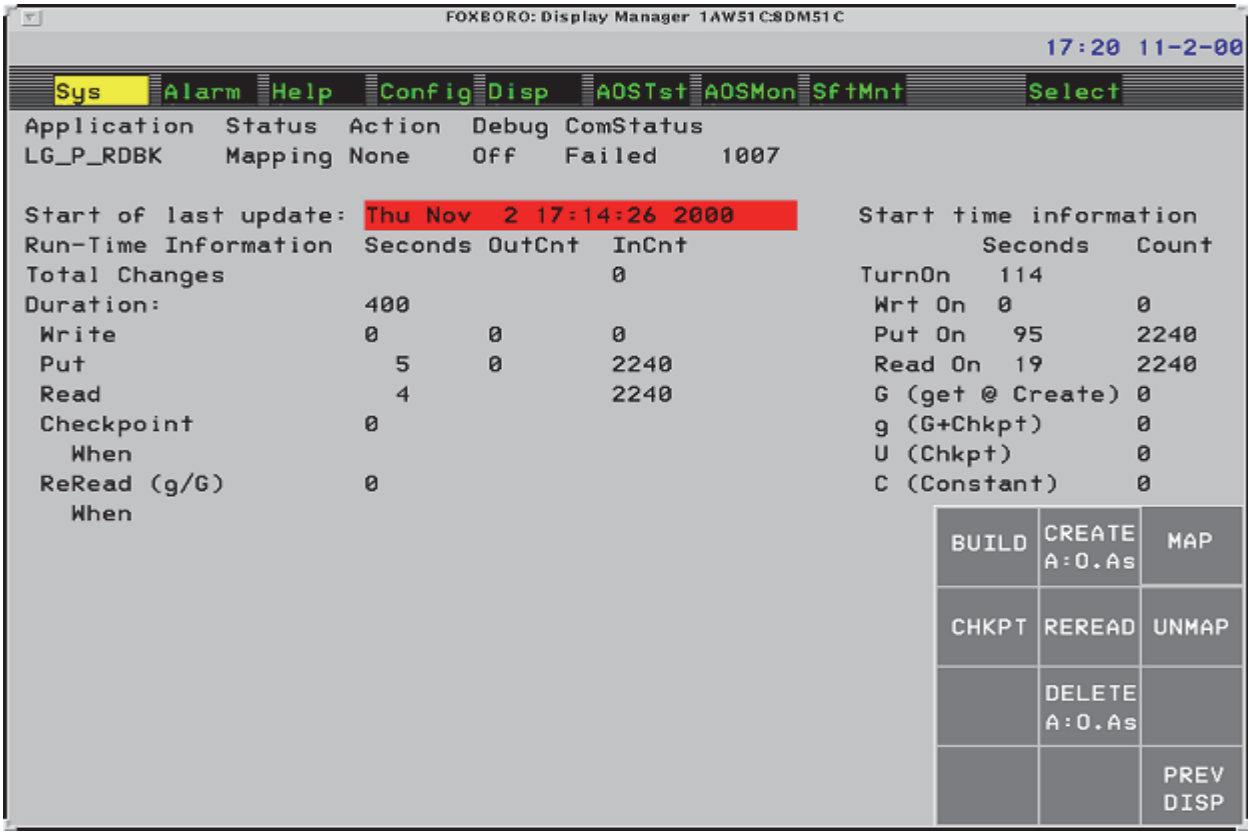


Figure 6-3. INI Application Monitoring Display (Failed Communications)

From this display the application can be rebuilt, the AOAs created, mapping started, checkpointing initiated, a re-read attempted, mapping stopped, and the AOAs deleted.

The status monitor displays buttons and their meanings are defined in Table 6-1

Table 6-1. INI Monitor Display Buttons

Button Label	Description
BUILD	Reads an application definition from a map file and stores the definition in the database.
CHKPT	Makes a checkpoint request of the INI.
CREATE AOS	Creates in memory the AOAs defined for the application. The definition is read from the database.
DELETE AOAs	Deletes the current application from memory.
MAP	Starts the mapping (transfer) of data between the local and Remote Stations.
PREV DISP	Raises the previous display.
REREAD	Makes a re-read request of the INI.
UNMAP	Makes an unmap request of the INI.

The status monitor displays fields and their meanings are defined in Table 6-2.

Table 6-2. INI Monitoring Display Data Fields

Field Name	Definitions
G (get @ Create)	Shows the number of **G** mappings required.
g (G+Chkpt)	Shows the number of **g** mappings required.
C (Constant)	Shows the number of **C** mappings required.
U (Chkpt)	Shows the number of **U** mappings required.
Application	Name of the application being mapped by an INI instance.
Action	Shows the currently requested action. The options are: ♦ **None** - No additional action has been requested. ♦ **CkptRqt** - Checkpoint has been requested. ♦ **ReReadG** - Rereading of the G/g mappings has been requested. ♦ **Remap** - Changes have been made to the application and these changes are being processed. ♦ **Unmap** - Stops the data transfer.
Checkpoint	Indicates the time in seconds spent performing **c** mapping operations.
Checkpoint When	Shows the time of the last checkpoint.
Debug	Indicates if debugging of the application is active. Debug output is written to the standard output and can be re-directed at startup.
Duration	Indicates the amount of time that the INI process required to update all of the AOAs that had changes.
Put	Indicates the time in seconds spent performing **P** mapping operations.
Put On (Count)	Shows the number of **P** mappings required.
Put On (Seconds)	Shows the time in seconds to open the read backs for the **P** mappings.
Read	Indicates the time in seconds spent performing **R** mapping operations.
Read On (Count)	Shows the number of **R** mappings required.
Read On (Seconds)	Shows the time in seconds to open the **R** mappings.
ReRead (g/G)	Indicates the time in seconds spent performing **G/g** mapping operations.
When	Shows the time of the last reread.
Wrt On (Seconds)	Shows the time in seconds to open the **W** mappings.
Wrt On (Count)	Shows the number of **W** mappings required.
Write	Indicates the time in seconds spent performing **W** mapping operations.

Table 6-2. INI Monitoring Display Data Fields (Continued)

Field Name	Definitions
`ComStatus`	Reports the status of the NetFoxAPI/NetAIM*API communications. There are three fields: ♦ The status of the communication link, ♦ The error communication error (.comerr attribute), and ♦ The communications state (.comst attribute). If the Status indicates either InProg or Created, the communication link status field shows Inactive. If the Status indicates Mapping, the communication link status field is Ok if there are no errors. If an error has occurred and the Status is Mapping, the communication link status field will show Failed. If there is a communications error, the NetFoxAPI/NetAIM*API error code will be displayed in the communications error field. If there is a communications error, the communication's error state of the application is shown in mnemonic form. The possible values are: **0 <blank>** **1 ...S** **2 ..P.** **3 ..PS** **4 .H..** **5 .H.S** **6 .HP.** **7 .HPS** **8 T...** **9 T..S** **10 T.P.** **11 T.PS** **12 TH..** **13 TH.S** **14 THP.** **15 THPS** **<blank>** signifies no error The first column indicates the status of the timeout timer. A dot (.) indicates no timeout. A **T** indicates a timeout. The second column indicates the status of the timeout timer. A dot (.) indicates no timeout. A **H** indicates that NetFoxAPI/NetAIM*API has returned an error code. The third column indicates the status of the timeout timer. A dot (.) indicates no timeout. A **P** indicates that there was an error during the previous scan. The fourth column indicates the status of the timeout timer. A dot (.) indicates no timeout. A **S** indicates that a soft NetFoxAPI/NetAIM*API error occurred. Soft errors may result in scrambled indices and, therefore, the error recovery mechanism must be invoked.

Table 6-2. INI Monitoring Display Data Fields (Continued)

Field Name	Definitions
`Start of last update`	Shows the time of the last INI AOA update. The format is: "**ddd mmm dd hh:mm:ss yyyy**" where: **ddd** is a three letter mnemonic for the day. **mmm** is a three letter mnemonic for the month. **dd** is the day number in the month. **hh:mm:ss** is the time in hours, minutes, and seconds. **yyyy** is the current year. This field will have a red background if there is an overrun, i.e., the duration exceeds the scan rate.
`Status`	Shows the status of the INI instance. The possible states are: ♦ **InProg** - the application is being created, but not all AOAs exist. ♦ **Created** - the application is fully created, but not yet transferring data. ♦ **Mapping** - data transfer for the application is in progress.
`TurnOn`	Shows the time in seconds required for the INI to achieve complete communications to the NetFoxAPI/NetAIM*API server.

Error Logs and Messages

The **D:\opt\aos\logs** directory holds error logs generated by the INI scripts. Table 6-3 associates the INI scripts to the log file they generate.

Table 6-3. Scripts and Corresponding Logs

Script	Debug Log Name	Error Log Files	Comments
`INIBUILDDB`		`aos_buildb`	
`INICHKPT`		`aos_chkpt`	
`INICREATE`		`aos_create`	
`INIDEBUG`		`aos_debug`	
`INIDELETE`		`aos_delete` `aos_disconnect`	Created if and only if **I** mappings are specified.
`INIMAP`	`aos_map.out`	`aos_map.err` `aos_connect`	The **-L** option allows a different directory to be specified. Created if and only if **I** mappings are specified in the Application Definition.
`INIMMAP`	`aos_map.out`	`aos_map.err` `aos_connect`	The **-L** option allows a different directory to be specified. Created if and only if **I** mappings are specified in the Application Definition.
`INIMODIFY`		`aos_modify`	
`INIREMAP`		`aos_remap`	
`INIREPORT`		`make_report`	

Table 6-3. Scripts and Corresponding Logs (Continued)

Script	Debug Log Name	Error Log Files	Comments
INIREREADG		aos_rereadg	
INIUNMAP		aos_unmap	
INIVERIFY		verifier	

During the start of mapping, the scripts will produce:

- **aos_parse.out** to hold error messages if there are problems checking the application's definition.
- **verifier** to hold error messages if there are problems verifying **I** mappings.

Operational

The INI70 executable prints error messages to stderr and debug messages to stdout. The scripts **INIMAP** and **INIMMAP** allow the user to specify the directory that will hold the debug and error logs. The file names will be aos_map.err and aos_map.out.

INIDEBUG can be used to start debug logging when an application is mapping.

Mapping the Application

Once the Application is created, connected, and verified, mapping may be started. There are several options:

- Use the **ini** executable directly if the environment variables are properly setup
- Use the **INIMAP** script to start data transfer between systems for a single application or the **INIMMAP** script to start data transfer between systems for multiple applications
- Use **AppObjSrv**

With the INI software release v5.00.00, the ini mapping is carried out by a Windows Service called iniMap (Figure 6-4). The iniMap service is installed during the installation of the INI70 software, but it has to be started before mapping can be performed.

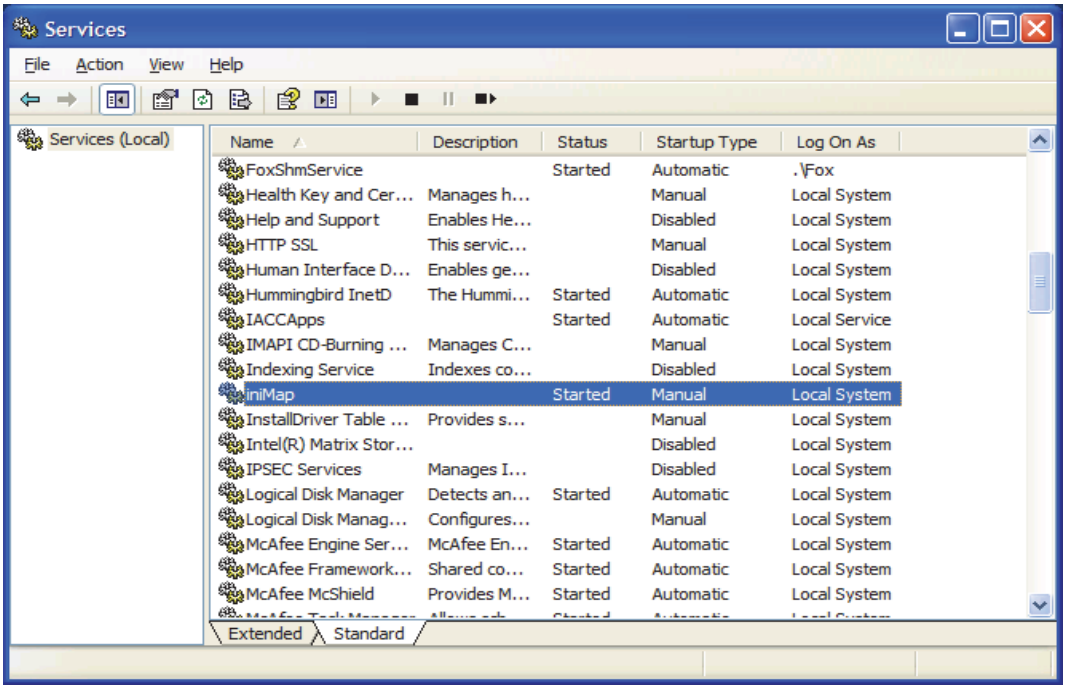


Figure 6-4. iniMap Service

In addition, debugging information from the mapping process cannot be obtained by Windows redirection as in:

```
/opt/aos/exe/ini.exe map APP 5 /opt/aos/data/defaultTrans >
../log/ini_map.out 2> ../log/aos_map.err
```

Instead the following should be used to capture debugging information:

`/opt/aos/exe/ini.exe map APP 5 /opt/aos/data/defaultTrans -D3` *(output to the screen)*
`/opt/aos/exe/ini.exe map APP 5 /opt/aos/data/defaultTrans +D3` *(output to the file /opt/aos/debug/aos_map_APP)*
`/opt/aos/exe/ini.exe map APP 5 /opt/aos/data/defaultTrans /D d:/opt/aos/logs` *(directs stdout to the d:/opt/aos/logs/aos_map.out file, and stderr to the d:/opt/aos/debug/aos_map.err file)*

On-line Changes

The INI software supports on-line changes to a mapping Applications. The general procedure is:

- Create a modified map file for the Application or Applications.
- Stop checkpointing on the mapping Applications. This must be done because the mapping process remembers the index for each **U/g** mapped Attribute and these indices change when a re-build occurs.
- Re-build the Application Definition in the database.
- Re-create the AOAs for the application. New AOAs are added, no AOAs are deleted, only **K** mappings have their values changed to match the value in the database, and no AOAs change data type.
- Repeat the stop checkpointing, re-build and re-map steps for each Application needing an update.

- Signal the instance to re-map when ready. The re-map step automatically restores checkpointing.

When re-mapping is triggered, the package:

- Re-reads the database to learn all of the attributes for all of the Applications being mapped. At this time, it learns the database index for each **U/g** mapped Attribute.
- Closes all **W** mappings.
- Adds any new **R** mappings.
- Adds read backs for any new **P** mappings.
- Opens the new set of **W** mappings.

Adding small numbers (hundreds) of points is fairly quick. Adding large numbers of points (thousands) can take a minute or more.

— NOTE

Since no updates are performed during the re-map process, be aware of the impact of on-line changes to the particular application.

Since re-building changes the indices of the **U/g** mappings, a re-map must be signaled after a re-build before resuming checkpointing. Failure to do this will result in the wrong data being stored into the database for **U** and **g** mapped Attributes.

Automated Re-mapping

Automated re-mapping involves running **INIREMAP**, see page 62, and answering the required questions. This tool automates the procedure described above once the questions are answered.

What follows is an example transcript. In this case, two instances of the package were running as shown here:

```
root  2146     1  0 14:36:00 pts/3  0:01 aosMonitor READ WRITE
root  2149  2146  0 14:36:01 pts/3  0:07 INI70 mmap READ WRITE -BT
/opt/INI70/data/BTRANS.V62
root  2372  2369  1 14:54:08 pts/3  0:14 INI70 map RDBK 2
/opt/INI70/data/BTRANS.V62
root  2369     1  0 14:54:07 pts/3  0:01 aosMonitor RDBK
```

In the following, the computer's output is in a bold, mono-spaced font, the user's entries are in the same font in italics and bold. Comments are interspersed in the text. They are the lines written in italics.

```
1AW70C# INIREMAP
The following Applications are mapping...
READ WRITE RDBK
```

The program lists all of the mapping applications.

```
Is there an Application that needs to be remapped? [y | N]
y
```

If the answer is n or N, the program will exit. Either y or Y will cause the script to proceed.

```
Enter the name of the application to rebuild or ctrl-C to exit:
READ
```

At this point, the user entered the name of a mapping Application.

```
Enter the name of the Map File for READ or ctrl-C to exit:
/opt/INI/examples/INI70/std/READ/data/READ.map
```

Generally, the full pathname should be entered. However, relative pathnames from the current directory will work.

```
Sending Skip Checkpoint request...
```

At this point, checkpointing is disabled. This is important because the program remembers the database indices for checkpointed variables and these indices will change in the next step.

```
Re-building Application Definition for READ...
```

Once this is complete, the checkpointing must not be re-enabled until the Application is re-mapped. Re-mapping causes the program to learn the new database indices.

```
Re-creating AOAs for READ...
```

Any attributes that were added to an application are created. Existing attributes are not changed. That is, no attributes are deleted, their data types are not changed, and only K mappings have their values set to the value stored in database.

```
Rebuild and re-create another Application? [y | N]
N
```

This query gives the user the chance to re-build and re-create other applications.

```
Current System Time==> Sat Jun  2 15:26:38 GMT 2011
Enter the name of the application to signal to re-map:
     The following applications are mapping:
READ
RDBK
```

List of the first application name on each instance of the package. Only the first Application name on a mapping instance of the package is checked for commands like re-map, unmap, and checkpoint.

```
READ
Re-mapping AOAs for READ...
```

Manual Re-Mapping

The manual procedure allows the user full control, but requires more typing and it requires the proper environment variable setup, page 32.The automated procedure semi-automates the process.

To re-map manually, do the following:

- Modify the workfile. This may require re-generation of the map file using **INIREPORT**, see page 62.
- Suspend checkpointing using **INI70 skipCP...** if any **U** mappings are specified in the application.
- Build the workfile using **INI70 buildDB...**
- Re-create the application so that any new objects and attributes are created: **INI70 recreate...**
- Trigger re-mapping, i.e., adapting to the new application definition, by using **INI70 remap...**. This operation will automatically resume checkpointing.

Message Forwarding

Status Monitoring

The processes that implement the Message Forwarding facility creates Application Objects to represent their status. Each instance of Message Forwarding (sending and receiving) creates an Application shown in Table 6-4.

Table 6-4. Message Forwarding - Monitoring Objects

Name	Purpose
<iaToQ_LN>:iaToQ.*	Reports the status of the iaToQ instance called <iaToQ_LN>, the Logical Name specified at iaToQ's startup.
<iaToQ_LN>:qToNet.*	Reports the status of the qToNet instance called created by the iaToQ instance called <iaToQ_LN>.
<netToIA_LN>:net-ToIA.*	Reports the status of the netToIA instance called <netToIA_LN>, the Logical Name specified at netToIA's startup.

The full Application Object Attribute definitions for Message Forwarding are listed in Table 6-5.

Table 6-5. Message Forwarding - Application Object Attribute Definitions

Application	Object	Attribute	Type	Definition
<iaToQ_LN>	iaToQ	NAME	STRING	Should match the <iaToQ_LN>.
		CLDON	OM_BOOL	If True, the child process that reads from the outgoing message queue is running. The child is represented by <iaToQ_LN:qToNet the
		CLDPID	OM_LNG_INT	Process Id of the child sending the message from the queue.
		MSGCNT	OM_LNG_INT	Counter that runs from 0 to 999999 and back as messages are placed in the outgoing queue.
	qToNet	NAME	STRING	Should match the <iaToQ_LN><pid>.
		CONNCT	OM_BOOL	Set if the child is connected to netToIA across the network.
				Counter that runs from 0 to 999999 and back as messages are taken from the outgoing queue and sent across the CSN.
<netToIA_LN>	netToIA	NAME	STING	Should match the <netToIA_LN>.
		CLDON	OM_BOOL	If True, the child process that reads from the network is running.
		CONNCT	OM_BOOL	Set if the child is connected to qToNet across the network.
		MSGCNT	OM_LNG_INT	Counter that runs from 0 to 999999 and back as messages are taken from the CSN and sent to control system destinations.

Error Logs and Messages

The executables support a debug option that may be applied during startup. Debug printing cannot be started during operation; it must be requested at startup.

The syntax of the debug options are defined in Table 6-6.

Table 6-6. Message Forwarding - Debug Options

String	Meaning
`-D<level>`	The program is run in the foreground with debug trace printed to stderr. The level number specifies the amount of debug. It can range from 1 to 5.
`+D<level>S[fileSize]`	The program is run in the background with debug trace printed to a file in the debug directory (**D:\opt\aos\MsgFwd\debug**). The file name will include the logical name of the Message Forward instance. The level number specifies the amount of debug. It can range from 1 to 5.

Message Relay

Status Monitoring

The processes that implement the Message Relay facility create Application Objects to represent their status. Each Message Relay sending instance creates an Application defined in Table 6-7.

Table 6-7. Message Relay - Monitoring Objects

Name	Purpose
`<msgRelay_LN>:iaToQ.*`	Reports the status of the portion of msgRelay that receives message of the Message Relay instance called <msgRelay_LN> where **<msgRelay_LN>** is the Logical Name specified at msgRelay startup.
`<msgRelay_LN>:qToNet.*`	Reports the status of the portion of msgRelay that sends messages over the CSN.
`<msgRelay_LN>:netToIA`	Reports the status of the portion of msgRelay that receives messages from the CSN and sends them to control system destinations.

The full Application Object Attribute definitions for Message Relay are listed in Table 6-8.

Table 6-8. Message Relay - Application Object Attribute Definitions

Application	Object	Attribute	Type	Definition
<iaToQ_LN	iaToQ	NAME CLDON CLDPID MSGCNT qToNet	STRING OM_BOOL OM_LNG_INT OM_LNG_INT STRING OM_BOOL	Should match the **<iaToQ_LN>**. If True, the child process that reads from the outgoing message queue is running. The child is represented by <iaToQ_LN:qToNet the Process Id of the child sending the message from the queue. Counter that runs from 0 to 999999 and back as messages are placed in the outgoing queue. Should match the <iaToQ_LN><pid>. Set if the child is connected to netToIA across the network. Counter that runs from 0 to 999999 and back as messages are taken from the outgoing queue and sent across the CSN.
<netToIA_LN>	netToIA	NAME CLDON CONNCT MSGCNT	STING OM_BOOL OM_BOOL OM_LNG_INT	Should match the **<netToIA_LN>** If True, the child process that reads from the network is running. Set if the child is connected to qToNet across the network. Counter that runs from 0 to 999999 and back as messages are taken from the CSN and sent to control system destinations.

Error Logs and Messages

The executables support a debug option that may be applied during startup. Debug printing cannot be started during operation; it must be requested at startup.

The syntax of the debug options are defined in Table 6-9.

Table 6-9. Message Relay - Debug Objects

String	Meaning
-D<level>	The program is run in the foreground with debug trace printed to stderr. The level number specifies the amount of debug. It can range from 1 to 5.
+D<level>S[file-Size]	The program is run in the background with debug trace printed to a file in the debug directory (**D:\opt\aos\MsgRelay\debug**). The file name will include the logical name of the Message Forward instance. The level number specifies the amount of debug. It can range from 1 to 5.

7. Maintenance

This chapter discusses the regular maintenance that may have to be done after the installation of the INI software.

Package Removal

Take the following steps to remove the INI package.

1. Stop all mapping applications on the Client by using INIUNMAP. Create a script as follows to generate a list of those processes to stop with INIUNMAP.

```
#!/bin/sh
cd /opt/aos/status
FILES=`ls INI51_map_* | grep -v '_monitor'`
for file in $FILES
do
/opt/aos/exe/testStatusLock $file | grep Locked >/dev/null 2>&1
status=$?
if [ $status = 0 ]
then
echo $file | awk -F_ '{ print $3 }'
fi
done
```

2. Use the script **D:\opt\aos\MsgFwd\scripts\stop_MsgFwd**, see page 72, on both the local and Remote Stations to stop message forwarding:

```
cd D:/opt/aos/MsgFwd/scripts
./stop_MsgFwd
```

3. Use the script **d:\opt\aos\MsgFwd\scripts\stop_MsgRelay**, see page 73,on either the Local Station or the Remote Station to stop message forwarding:

```
cd D:/opt/aos/MsgRelay/scripts
./stop_MsgRelay
```

4. Use Add or Remove Programs or Programs and Features to unininstall the INI70 software.
   a. From the Windows **Control Panel**.
   b. Select **Add or Remove Programs or Programs and Features**.
   c. Select the **INI70 with Message Services** item.
   d. Select the **Remove** button.
5. Remove the **AOSINI** line from the `D:\usr\fox\bin\fox_apps.dat` file.
6. Remove the **go_AOSIN0.ksh** script from the `D:\usr\fox\bin` directory.
7. Remove the `D:\opt\aos` directory from both workstations. (Do not remove this directory if the AOS software is installed.)

Upgrades

Day 0 Upgrades

In the case of a Day 0 upgrade, preserve the INI configuration. The discussion on page 58 in the part of the INI Package Upgrades section - explains how to preserve the configuration information.

The INI package should then be reinstalled from the original or replacement media. Frequently, the most recent version of the INI is available upon request so please investigate obtaining the latest materials by contacting your Field Service or Account Representative.

non-Day 0 Upgrades

Other than the file **D:\usr\fox\bin\fox_apps.dat** file, the INI software should not be impacted by a non-Day 0 upgrade. However, it would be wise to backup the **D:\opt\aos** directory and the information in the database.

The discussion on INI Package Upgrades explains how to preserve the configuration information.

INI Package Upgrades

In all cases, consult the specific upgrade instructions that come with the upgrade materials. The following information is generic.

General

The INI software can be upgraded without rebooting the hosting workstation. However, the INI package must be shutdown first.

The steps in an upgrade are:

1. Backup configuration data.
2. Stop the package.
3. Load the new files.
4. Install the new files.
5. Restore the configuration data.
6. Restart the package.

Saving Configuration Data

1. Preserve the system file **C:\windows\system32\drivers\etc\services**.
2. Preserve the startup file **D:\usr\fox\bin\fox_apps.dat**.
3. Preserve the startup file **D:\usr\fox\bin\go_INI70.ksh**.
4. The configuration file used by the Message Forwarding component of the INI package that may be replaced during an upgrade is:
   - **D:\opt\aos\MsgFwd\data\FileOfNames** (The configuration file for the Message Forwarding service.)
5. The configuration files used by the Message Relay component of the INI package that may be replaced during an upgrade is:
   - **D:\opt\aos\MsgFwd\data\FileOfProxies**

- **D:\opt\aos\MsgFwd\data\FileOfTargets**

An database stores the configuration information related to the Data Transfer portion of the INI package. The information in this database can be converted back into map files by the use of the **INIREPORT** script documented on page 62.

Most upgrades will not require this step, but any upgrade that alters the database will.

Stopping the INI Software

To halt the INI software, use the **INIUNMAP** command to unmap the INI instances currently running.

1. Use the following example to create a script to see a list of the current INI instances.

```
#!/bin/sh
cd /opt/aos/status
FILES=`ls INI51_map_* | grep -v '_monitor'`
for file in $FILES
do
/opt/aos/exe/testStatusLock $file | grep Locked >/dev/null 2>&1
status=$?
if [ $status = 0 ]
then
echo $file | awk -F_ '{ print $3 }'
fi
done
```

2. Once the INI instances are stopped, suspend Message Forwarding, on each machine, by issuing the command:

```
D:/opt/aos/MsgFwd/scripts/stop_MsgFwd
```

3. Once the INI instances are stopped, suspend Message Relay facility by issuing the command:

```
D:/opt/aos/MsgFwd/scripts/stop_MsgRelay
```

Loading INI Package

Install the INI software per the instructions found in “INI70 Software Installation” on page 20.

Restoring Configuration Information

To restore configuration information, please do the following:

- Confirm the contents of **D:\usr\fox\bin/fox_apps.dat**. Do not simply restore the previous copy.
- Confirm the contents of the **C:\windows\system32\drivers\etc\services** file. Do not simply restore the previous copy.
- Restore the **FileOfNames** file used by Message Forwarding.
- Restore the **FileOfProxies** and **FileOfTargets** files used by the Message Relay facility.
- Rebuild the database from the map files if necessary.

Restart the INI Package

The simplest restart mechanism would be to reboot.

Quick Fixes

After non-INI software related Quick Fixes are installed, the following should be checked:

- The **D:\usr\fox\bin\fox_apps.dat** file should contain the INI Package startup commands.
- The file **C:\windows\system32\drivers\etc\services** file should contain the Message Forwarding and Message Relay service definitions.
- The command **show_params** should show the same information as before the Quick Fix was installed.

8. Command Usage

This chapter discusses the various scrips and executables included as part of the INI software.

Map File Building

GatherWorkFiles

Copies the **<cpLbug>** directory for each control station to a specified directory.

SYNOPSIS

```
GatherWorkFiles [<cpLbug> ... <cpLbug>]
```

DESCRIPTION

The script **GatherWorkFiles** copies the workfile directory for a specified set of Control Stations to **D:\opt\aos\wf**. Once collected, these workfiles can be used to build the index of blocks for the system using **mkBlkIndex**. The index can be used by **mkINIMapFile** to create map files.

The rules to specify Control Station letterbugs are:

- If Control Station letterbugs are specified on the command line, those are the only Control Stations whose workfiles are copied.
- If no Control Station letterbugs are specified on the command line, the script uses those letterbugs specified in the file **D:\opt\aos\data\CSs** if it exists
- If no Control Station letterbugs are specified and the CSs file does not exist, the contents of **D:\etc\cplns** are used as the list of letterbugs.

This script needs to be run on a workstation that resides on the "remote" network. It cannot gather workfiles over the second Ethernet port.

— NOTE

Do not gather the workfile of a control station while that station is being edited using the ICC or ICCAPI. Doing so will result in copying an inconsistent file and lead to an incorrect map files.

EXAMPLE

Typical usage is as follows:

```
cd /opt/aos/scripts
./GatherWorkFiles
../exe/mkBlkIndex ../wf/*/*.wf | sort -o ../wf/BlockIndex
../exe/mkINIMapFile ../wf/BlockIndex -d ../tmp <../data/mapDef
```

INI Operation

The INI Operation scripts are:

INIBUILDDB - Builds the database from a map file.

INICHKPT - Initiates the checkpoint operation.

INICREATE - Creates the Application.

INIDEBUG - Initiates debug printing on a mapping application.

INIDELETE - Deletes the Application from CSA and from the OM's Shared Memory and, optionally, disconnects the AOAs from their I mapping targets.

INIMAP - Starts data transfer between systems for a single application.

INIMMAP - Starts data transfer between systems for multiple applications.

INIMODIFY - Merges a new map file and the current database eliminating deleted objects and adding new ones.

INIRECREATE - Creates any new attributes for an Application.

INIREMAP - Causes INI software to alter its mappings to conform to an update in the database.

INIREPORT - Creates a map file from the contents of the database.

INIREREADG - Initiates an update of g mappings.

INIUNMAP - Stops data transfer.

INIVERIFY - Verifies that all I mapping connects have been made.

SYNOPSIS

```
INIBUILDDB <app_data_dir> <applicationName> <map_file>
INICHKPT <applicationName>
INICREATE <applicationName> [skipCSA] [forceCSA]
INIDEBUG <applicationName> <debugCount>
INIDELETE <applicationName> [skipCSA] [forceCSA] [disconnect]
INIMAP <applicationName> <scanRate> [-AN <an_init.cfg fullpathname>]
     [-CF <chkptPeriod>] [-NOP] [-L <logfiledir>] [-M <mapfiledir>] /D
<output>
INIMMAP <applicname> [<applicname>...] [-AN <an_init.cfg fullpathname>]
     [-SR <map_rate>] [-Connect] [-CF<checkpoint_rate>] [-NOP]
     [-L <logfiledir>] [-M <mapfiledir>] [/D <output>]
INIMODIFY <app_data_dir> <applicationName> <map_file> <old_map_file>
INIRECREATE <applicationName>
INIREMAP
INIREPORT <app_data_dir> <applicationName> <outfile> [<object>]
INIREREADG <applicationName>
INIUNMAP <applicationName>
INIVERIFY <applicationName> [<verbose>] [<Galaxy>]
```

where:

<object> - is the name of an object in an Application.

-AN <an_init.cfg fullpathname> - specifies the full path name of the **an_init.cfg** file to use. By default, the an_init.cfg in **d:\opt\aos\data** or, if specified, the directory given by the **-M** option is used.

-CF- <chkptPeriod> specifies the checkpoint period in seconds.

-Connect - specifies that the script should connect the **I** mappings.

disconnect - specifies that the script should break **I** mappings.

forceCSA - specifies that the script should attempt to force an update to CSA.

L <logfiledir> - specifies the directory that will hold the log files.

-M <mapfiledir> - specifies the directory that will hold the map files that is generated.

-NOP - specifies that read backs of **P** mappings should be skipped.

skipCSA - specifies that CSA registration/de-registration should be skipped.

-SR <map_rate> - specifies the scan rate (mapping rate) of the Application.

verbose - specifies that the script should print a trace of its actions.

<application_data_dir> - specifies the directory that holds the map file.

<applicationName> - specifies the Application that is being manipulated.

<debugCount> - specifies the number of cycles that debug will be turned on.

<map_file> - specifies the name of the map file to process.

<old_map_file> - specifies the name of the old map file.

<outfile> - specifies the name of the file to hold the output.

<scanRate> - specifies the scan rate (mapping rate) of the application.

<Galaxy> - specifies the galaxy name.

/D <output> - specifies the files for stdout (*.out) and stderr (*.err) messages. Further examples of using the /D option are defined in Table 8-1 on page 66.

DESCRIPTION

The **aos** executable implements the data transfer portion of INI package, i.e.,

- Building an Application Object Database,
- Creating an application's objects in memory,
- Mapping an application's objects to and from their appropriate locations, and
- Recording specific attribute values into the Application Object Database.

Several commands (**INIBUILDDB**, **INICREATE**, **INIMAP**, **INICHKPT**, **INIUNMAP**, **INIDELETE**, and **INIREREADG**) exist to simplify the use of these services. One additional command (**INIDEBUG**) exists to facilitate problem resolution. The **INIREPORT** and **INIMODIFY** commands exist to allow user maintenance of the database contents. The **INISKIPCP** and **INIRESUMECP** commands facilitate the re-mapping.

INIBUILDDB

INIBUILDDB <app_data_dir> <applicationName> <map_file>

This command loads an application, **<applicationName>**, and all of its component AOAs into the database given a map file, **<map_file>**, and the map file location, **<app_data_dir>**. The application loader creates the **INI** schema if necessary.

Multiple applications are allowed in the database at the same time.

The application loader verifies that each map file entry contains:

- Well-formed AOA name, with max lengths of 12 for object and 6 for attribute.
- CBP name length <= 32.
- Mapping Type matching one of those described on page 95.
- Data Type matching one of those described.
- Boolean values must be 0 or 1.

INICHKPT

INICHKPT <applicationName>

This command signals the mapping process to all records that can be updated by AOAs belonging to the application. An AOA is checkpointed by writing its shared variable's value to the database. The application must be mapping for checkpointing to occur. Only AOAs specified with a **U** mapping are recorded in the database, i.e., checkpointed.

INICREATE

INICREATE <applicationName> [**skipCSA**] [**forceCSA**]

This command creates a shared variable for each AOA in the application database. It initializes each AOA that is to be updated to its last checkpointed value and each constant AOA to its value. The Application definition database must already have been loaded into the database.

During the creation of AOAs, the INI software may create 12-byte float variables to hold the high scale, low scale, change delta. For each AOA with an attribute of length 5 whose first 3 characters are HSC, the INI software creates a 12-byte float named "R[suffix of HSC attribute]". These 12-byte float variables cannot be defined in the user map file. The AOS software automatically maintains their values as the HSC, LSC, or DELT parameters change.

The **skipCSA** option prevents **INICREATE** from registering the Application and Objects with CSA. The **forceCSA** option should be used if there are problems registering from CSA.

INIDEBUG

INIDEBUG <applicationName> <debugCount>

This command controls debugging for the application. Mapping and checkpointing information is printed to the standard output device. The debug output is treated as a circular queue and will contain no more than debug information from the last 100 cycles of execution. A **<debugCount>** value of 0 (zero) turns debugging off, a **<debugCount>** of 32767 turns debugging on permanently, and a value between 1 and 100 turns debugging on for that specific number of iterations.

INIDELETE

INIDELETE <applicationName> [disconnect] [skipCSA] [forceCSA]

This command deletes each application object attribute for the specified application, <applicationName>. If the disconnect option is specified, the Application Objects/CBP connections associated with Mapping Types, **I**, are disconnected before Application Objects are deleted.

The **skipCSA** option prevents **INIDELETE** from un-registering the application and Objects with CSA. The **forceCSA** option should be used if there are problems un-registering from CSA.

INIMAP

INIMAP <applicationName> <scanRate> [-AN<an_init.cfg fullpathname>] [-Connect] [-CF <chkptPeriod>] [-NOP] [-L <logfiledir>] [-M <mapfiledir>] [/D <output>]

This command starts mapping for the specified application, **<applicationName>**, with the mapping rate specified by the second argument, **<scanRate>**.

The INI executable requires an **an_init.cfg** file.

The INI executable assumes that the file is in the current directory by default. If the file cannot be found, the INI executable does not start and a message will be placed in the **aos_map.err** file in the logs directory. The logs directory is either **D:\opt\aos\logs** or one specified by the **-L** option.

The **INIMAP** script sets the current directory before it runs the INI executable. If the **-M** option is specified, the executable will be run from that directory. If the **-M** option is not specified, the executable will be run from the **D:\opt\aos\data** directory.

The **-AN** option can be used to override the default location of the an_init.cfg file and the location specified by the **-M** option by directly and fully specifying the path name of the an_init.cfg file. The table below shows possible invocations.

If the **-Connect** option is specified, INIMAP command connects any **I** mapped Application Object Attributes to their assigned CBP. The connection is made by using the program connect. Connect uses the ICCAPI Driver Task to write the name of the AOA into the CBP.

If the **-Connect** option is not specified, INIMAP command first verifies that any Application Objects/CBP connections with Mapping Types of **I** are correctly established. If they are not connected properly, mapping will not be allowed to start.

The **-CF** option allows the operator to specify a checkpoint period. The checkpoint period is the frequency with which the values in **U** mapped variables will be checked for changes and written to the database.

The **-NOP** option is used to disable **P** mapping readback. It is required when the target device is a Spectrum Master Gateway. These devices do not support readback on all possible targets.

The **-L <logfiledir>** option specifies where the log files should be placed. The default directories is **D:/opt/aos/logs**. The path name must be absolute since the script changes directories.

The **-M <mapfiledir>** option is used to specify a particular location used to store the map file generated by the script. The default directories is **D:\opt\aos\data**. The path name must be absolute since the script changes directories.

The **/D <output>** option is used to create files for stdout (*.out) and stderr (*.err) messages. Where <output> can be a directory name, a filename or prefix of a filename. The file name can be defined with an absolute or relative pathname.

For example, the following map command will produce two output files in the **D:\opt\aos\logs** directory with the default prefix of aos_map.

```
/opt/aos/exe/aosmap MyApp2 /opt/aos/data/defaultBTrans /D
/opt/aos/logs
```

Examples of using the /D option are defined in Table 8-1.

Table 8-1. Examples Using the INIMAP/INIMMAP /D Option

/D <output>	stdout	stderr
`/opt/aos/logs`	/opt/aos/logs/aos_map.out	/opt/aos/logs/aos_map.err
`../logs`	../logs/aos_map.out	../logs/aos_map.err
`/opt/fox-ind/MyApp/MyApp.log`	/opt/foxind/MyApp/MyApp.log	/opt/foxind/MyApp/MyApp.log
`/opt/foxind/MyApp` (Where MyApp is a directory.)	/opt/foxind/MyApp/aos_map.out	/opt/foxind/MyApp/aos_map.err
`/opt/foxind/MyApp/App1` (Where App1 is a prefix of the output files.)	/opt/foxind/MyApp/App1.out	/opt/foxind/MyApp/App1.err

Some possible INIMAP startup options are described in Table 8-2.

Table 8-2. Possible Startup Options Using INIMAP

Description	Command
To startup with files in default locations: ♦ **PATH** includes **/opt/aos/scripts** ♦ **an_init.cfg** in **/opt/aos/data** ♦ **logs** assigned to **/opt/aos/logs** ♦ **map** file placed in **/opt/aos/data**	`INIMAP READ 2`
To startup in the directory holding the **an_init.cfg** file and specifying a specific **logs** directory based on the startup location.	`cd /opt/aos/examples/ini/READ/data` `INIMAP READ 2 -L ‘pwd‘/../logs -M ‘pwd‘`

Generalized Startup - Does not assume any **PATH** setting or directories.	`/opt/aos/scripts/INIMAP READ 2 \`
	`-AN /opt/aos/examples/ini/READ/data \`
	`-M /opt/aos/examples/ini/READ/data \`
	`-L /opt/aos/examples/ini/READ/logs`

Of course, the application must be created by **INICREATE** step before **INIMAP** will start mapping.

INIMMAP

INIMMAP <applicname> [**<applicname>...**] [**-AN <an_init.cfg fullpathname**] [**-SR <map_rate>**] [**-Connect**] [**-CF<checkpoint_rate>**] [**-NOP**] [**-L <logfiledir>**] [**-M <mapfiledir>**] [**\D <output>**]

This command starts mapping for the specified applications, **<applicationName>**.

The INI executable requires an **an_init.cfg** file.

The INI executable assumes that the file is in the current directory by default. If the file cannot be found, the INI executable does not start and a message will be placed in the **aos_map.err** file in the **logs** directory. The **logs** directory is either **D:\opt\aos\logs** or one specified by the **-L** option.

The INIMAP script sets the current directory before it runs the INI executable. If the **-M** option is specified, the executable will be run from that directory. If the **-M** option is not specified, the executable will be run from the **D:\opt\aos\data** directory.

The **-AN** option can be used to override the default location of the **an_init.cfg** file and the location specified by the **-M** option by directly and fully specifying the path name of the **an_init.cfg** file. The table below shows possible invocations.

The default mapping/scan rate is 2 seconds. An alternate mapping rate (in seconds) specified by the **-SR** option.

If the **-Connect** option is specified, INIMAP command connects any **I** mapped the Application Object Attributes to their assigned CBPs. The connection is made by using the program **Connect**. **Connect** uses the ICCAPI Driver Task to write the name of the AOA into the CBP.

If the **-Connect** option is not specified, INIMAP command first verifies that any Application Objects/CBP connections with Mapping Types of **I** are correctly established. If they are not connected properly, mapping will not be allowed to start.

The **-CF** option allows the operator to specify a checkpoint period. The checkpoint period is the frequency with which the values in **U** mapped variables will be checked for changes and written to the database.

The **-NOP** option is used to disable **P** mapping readback. It is required when the target device is a Spectrum Master Gateway. These devices do not support readback on all possible targets.

The **-L <logfiledir>** option specifies where the log files should be placed. The default directories is **D:\opt\aos\logs**. The path name may be relative or absolute. If a relative path name is used, be sure that the script is invoked from the correct location.

The **-M <mapfiledir>** option is used to specify a particular location used to store the map file generated by the script. The default directories is **D:\opt\aos\data**. The path name may be relative or absolute. If a relative path name is used, be sure that the script is invoked from the correct location.

The **/D <output>** option is used to create files for stdout (*.out) and stderr (*.err) messages. Where <output> can be a directory name, a filename or prefix of a filename. The file name can be defined with an absolute or relative pathname. For example, the following map command will produce two output files in the **D:\opt\aos\logs** directory with the default prefix of aos_map.

```
/opt/aos/exe/aosmap MyApp2 /opt/aos/data/defaultBTrans /D
/opt/aos/logs
```

Further examples of using the /D option are defined in Table 8-1 on page 66.

Some possible INIMMAP startup options are described in Table 8-3.

Table 8-3. Possible Startup Options Using INIMMAP

Description	Command
To startup with files in default locations: ♦ **PATH** includes **/opt/aos/scripts.** ♦ **an_init.cfg** in **/opt/aos/data.** ♦ **logs** assigned to **/opt/aos/logs.** ♦ **map** file placed in **/opt/aos/data.** ♦ default scan rate of 2 seconds.	`INIMMAP READ WRITE RDBK`

To startup in the directory holding the **an_init.cfg** file and specifying a specific **logs** directory based on the startup location and a scan rate of 4 seconds.	`cd /opt/aos/examples/ini/READ/data`
	`INIMMAP READ WRITE RDBK -SR 4`
	`-L ‘pwd‘/../logs -M ‘pwd‘`

Generalized Startup - Does not assume any **PATH** setting or directories, but uses the default scan rate of 2 seconds.	`/opt/aos/scripts/INIMMAP READ WRITE RDBK \`
	`-AN /opt/aos/examples/ini/READ/data \`
	`-M /opt/aos/examples/ini/READ/data \`
	`-L /opt/aos/examples/ini/READ/logs`

Of course, the application must be created by **INICREATE** step before **INIMMAP** will start mapping.

INIMODIFY

INIMODIFY <app_data_dir> <applicationName> < mapFile> <oldMapFile>

This option determines objects that were added to or removed from an existing map file, **<oldMapFile>**, which can be generated with **INIREPORT** option, and the latest user-edited map file. It then creates a master map file and replaces the current database contents with the contents of this master map file.

INIRECREATE

INIRECREATE <applicationName> [**skipCSA**] [**forceCSA**]

This command creates a shared variable for each AOA in the Application Database that does not already exist. The values of existing AOAs are not changed by this command.

It is used as part of the overall re-mapping operation.

The **skipCSA** option prevents **INIRECREATE** from registering the Application and Objects with CSA. The **forceCSA** option should be used if there are problems registering from CSA.

INIREREADG

INIREREADG <applicationName>

This option will cause the re-reading of all **G** Mapping Types (one-shot reads), when executed during the mapping operation.

INIREMAP

INIREMAP

Unlike the other INI Operation scripts, this script is interactive. It walks the user through the process of re-mapping one or more applications. An annotated transcript can be found on page 52.

INIREPORT

INIREPORT <appDataDir> <applicname> <outMapfile> [**<object>**]

This option will generate a new map file from the current contents of the Application Object database, and will place that map file in **<outMapfile>** in the **<appDataDir>** directory. If present, **<object>** is the Application Object for which Application Object database information is retrieved. If not present, all Application Object information for **<applicationName>** is retrieved.

INIUNMAP

INIUNMAP <applicationName>

This command turns off mapping for the specified application. If the multiple map option is being used, the Application name should be the name of the first application on the mapping command line.

INIVERIFY

INIVERIFY <applicationName> [**<verboseFlag>**] [**<Galaxy>**]

This command kicks off a task that verifies the connections between **I** mapped **CBPs** and their corresponding AOAs in the Application Object database. If set to **Y**, the <verboseFlag> causes additional informational messages to appear.

go_INI70

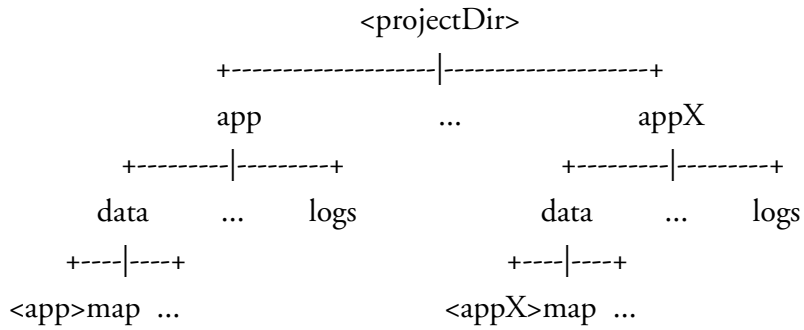
Starts the specified INI applications.

SYNOPSIS

```
go_INI70 <projectDir> <app> [<app>...]
```

DESCRIPTION

The **go_INI70** script assumes the following directory structure:



<projectDir> - is the directory that holds all the project specific files related to the INI software.

<app> ... <appX> - are directories that hold files for a specific application.

<projectDir>/<app>/<data> - holds the **<app>.map** file.

<projectDir>/<app>/<logs> - holds the log files.

If the subdirectories of the specified **projectDir** do not exist, it creates them.

Despite the existence of multiple logs, if **INI** is mapping more than one Application all of the messages go into the log file for the first Application on the command line.

The data directory must contain the **an_init.cfg** file used to tell the **INI** program where to get its data.

Use **lsApp INI** to get a complete list of INI applications.

The applications are started with a mapping (scan) rate of 5 seconds.

Database Query Scripts

The scripts used to query the database are:

lsApp, **lsAppObj**, **lsAppObjAttr** and **lsAppObjTypes**.

NAME

lsApp - Lists the applications found in the database.

lsAppObj - Lists the objects found in a specific application.

lsAppObjAttr - Lists the attributes of the specified object in the specified application.

lsAppObjTypes - Lists the object types of the objects in the database.

SYNOPSIS

```
lsApp [ INI | AOS ] [-h]
lsAppObj <appName> [-h]
lsAppObjAttr <appName> <objName> [-h]
lsAppObjTypes [-h]
```

DESCRIPTION

By default, **lsApp** lists all applications found in the database. If either **INI** or **AOS** is specified, it will list only those specific types of applications.

By default, **lsAppObj** lists all objects that belong to the specified application.

By default, **lsAppObjAttr** lists the attributes of the specified object in the specified application.

By default, **lsAppObjTypes** lists the object types found in the database.

The [**-h**] option prints a brief description of how to use these commands.

Message Forwarding

MsgStatus

MsgStatus - Verifies the status of the **iaToQ** and **qToNet** processes.

SYNOPSIS

```
MsgStatus
```

DESCRIPTION

The **MsgStatus** utility script that can be used on the Remote Station to verify that the **iaToQ** and **qToNet** processes are running.

stop_MsgFwd

Stops the execution of any Message Forward components.

SYNOPSIS

```
stop_MsgFwd
```

DESCRIPTION

The **stop_MsgFwd** utility script can be used on either the Remote Station or Local Station. This script will stop all components of the Message Forwarding component of the INI package that are running on the station that runs this utility.

testAlm

Sends a test message to the Message Forwarding software.

SYNOPSIS

```
testAlm <logicalName> <A | K | N>
```

where:

<logicalName> - One of the group names created in the **FileOfNames** file on the Remote Station. This parameter is required. This name could be logical names for stations and supplied programs, e.g., the Historian and the display managers are assigned using the System Configurator.

<A | K | N> - (**A**) Into Alarm; (**K**) Acknowledged; or (**N**) Return to Normal.

DESCRIPTION

The **testAlm** utility script is used for testing from the Remote Station. This script will send the Local Station an Alarm, Alarm Acknowledgment, or a Return-To-Normal message.

Message Relay

stop_MsgRelay

Used to stop all copies of Message Relay on the machine and their partners on the other station.

SYNOPSIS

```
stop_MsgRelay [<msgRelayName>]
```

DESCRIPTION

The **stop_MsgRelay** script is used stop all instances of **msgRelay** working with the current machine or one particular instance and its mate.

The **<msgRelayName>** option allows the user to specify the **msgRelay** instance to stop.

Miscellaneous

tellInetd

Notifies the system that the Message Forwarding is active.

SYNOPSIS

```
tellInetd
```

DESCRIPTION

The **tellInetd** utility script can be used on the Local Station during installation of the Message Forwarding or Message Relay component of the INI package. This script will notify the Local Station that the service **netToIA** has been installed and is ready to be invoked should communication from the Remote Station Message Forwarding component of the INI package processes arrive.

Executables

calcAppSize

Calculates the proper setting for **OM_NUM_OBJECTS**.

SYNOPSIS

```
calcAppSize [-v] [-A] [-h] [-D[1|2|3|4|5]] mapFile...
```

DESCRIPTION

The utility **calcAppSize** reads the specified map files and produces, by default, a single value that is the minimum setting for **OM_NUM_OBJECTS**.

The options are:

-v which lists each file and its usage.

-A which forces the output to count AOAs not **OM_NUM_OBJECTS**.

-h which generates a simple help.

-D[**1**|**2**|**3**|**4**|**5**] which turns on debugging at the specified level.

Experimentation has shown that the following rule approximates the correct setting for **OM_NUM_OBJECTS** and its high limit:

1. If the application does not use Application Object Alarming, take the number of lines in the map file and divided by 2.5 as the increment to **OM_NUM_OBJECTS**.
2. If the application uses a large amount of Application Object Alarming with text descriptors, take the number of lines in the map file as the increment to **OM_NUM_OBJECTS**.

In either case, add 1250, the space that the system needs, as the new upper limit and the new **OM_NUM_OBJECTS** value.

In addition, one should be aware that **calcAppSize** program calculates only the space required to hold the Application Object Attributes. Additional space is required to hold the OM lists that may be generated by applications that use these values. Each OM list requires 64 bytes for a header and 64 bytes for each variable. Using this information plus the fact that each **OM_NUM_OBJECTS** increment make 350 bytes available, one can estimate the additional increase required to support these lists.

EXAMPLE

If

```
calcAppSize ../data/CDUVDU.map
```

is entered the output might be:

```
5742
```

which could be used as a setting for **OM_NUM_OBJECTS** since it includes the base level of 1250 that the system expects for its own use.

If

```
calcAppSize -v TESTAPPLICTN.map CDUVDU.map EXAMPL.map BITTWIDDLE.map
     OC_WDR.map
```

is entered the output might be:

```
                                AOA Lines    OM Obj  Cnt  Ratio
TESTAPPLICTN.map          :           1           40     0.03
CDUVDU.map                :       12256         4492     2.73
EXAMPL.map                :         191          108     1.77
BITTWIDDLE.map            :          26          120     0.22
OC_WDR.map                :        1342         1301     1.03
Total for AppObjSrv       :       13816         6059     2.28
Total for OM_NUM_OBJECTS  :                     7309
```

This verbose listing shows what each application needs, the total for all Applications, and the overall number which includes the 1250 required by the system.

If

```
cd /opt/aos/examples/ini
calcAppSize -v -A */data/*.map
```

is entered the output might be:

```
                                  AOA Lines  Attributes  Lines/Attr
ADD_ONE_RWP.map             :            16         120        0.13
ADD_ONE_RWP_BIGGER.map      :            32         150        0.21
CAST.map                    :           144         262        0.55
ICC.MAP.map                 :             1          98        0.01
LG_P_RDBK.map               :          2240        4290        0.52
LG_P_RDBK_BIGGER.map        :          4480        8490        0.53
LG_P_W_RDBK.map             :          3856        7320        0.53
PUTONLY.map                 :             1          98        0.01
RDBK.map                    :            32         150        0.21
READ.map                    :             1          98        0.01
READCBP.map                 :             1          98        0.01
STRING.map                  :             2          99        0.02
TRIGGERED.map               :            69         187        0.37
WRITE.map                   :             2         106        0.02
WRITEONLY.map               :             1          98        0.01
Total for AOS usage         :         10878       21664        0.50
```

This verbose listing shows what each application needs, the total for all Applications, and the ratio of lines in the file to attributes created. The number is less than 1 because a certain number of attributes are created even if they are not defined in the map file.

INI70

Represents remote data as if they were local.

SYNOPSIS

```
INI70 builddb    <applicationName> <map text file> [-log[Transactions]]
INI70 checkpoint <applicationName>
INI70 create     <applicationName> [forceCSA] [skipCSA]
INI70 debug      <applicationName> <value>
INI70 delete     <applicationName> [forceCSA] [skipCSA]
INI70 map       <applicationName> <scan rate> <trans table>
                         [-CF <checkpoint frequency(seconds)>]
                         [- monitored]
                         [-NOP]
                         [-AN <path to an_init.cfg>]
                         [-WDT <timeout(sec)>]
                         [-RD <delay before recovery attempt(seoncds)>]
                         [-MapAlias <aliasMapName>]
                          [/D <output>]
INI70 mmap      <applicationName> <applicationName1>
[<applicationName2>...]
                         [-SR <scan rate>]
                         [-BT <trans table>]
                         [-CF <checkpoint frequency(seconds)>]
                         [-monitored]
                         [-NOP]
                         [-AN <path to an_init.cfg>]
                         [-WDT <timeout(sec)>]
                         [-RD <delay before recover attempt(sec)>]
                         [-MapAlias <aliasMapName>]
                          [/D <output>]
INI70 modify     <applicationName> <map text file>
INI70 parse      <applicationName>
INI70 recreate   <applicationName> [forceCSA] [skipCSA]
INI70 remap      <applicationName>
INI70 rereadg    <applicationName> <object name>
INI70 resumeCP   <applicationName>
INI70 skipCP     <applicationName>
INI70 unmap      <applicationName>
```

where

<applicationName> is the name of the Application to be manipulated.

<map text file> is the name of the map file used to create an Application Definition in the database.

<scan rate> defines how often the mapping process transfers data.

<trans table> is the name of the file that defines the bit map mnemonics.

<value> is the number of mapping cycles that debug should be printed.

-AN <an_init.cfg directory> specifies the directory that holds the an_init.cfg directory. This option exists for the AOS package, but is not used.

-BT <trans table> specifies the bit translation table to use.

-CF <chkptPeriod> specifies the checkpoint period in seconds.

-NOP Disables P mapping read-backs. Required for Spectrum Master Gateway.

-SR <map_rate> specifies the scan rate (mapping rate) of the Application.

-WDT <timeout(sec)> specifies the amount of time to wait before assuming a communication failure.

-MapAlias <newAppName> specifies the name used for the status files of the application. Use of this option allows both an aos and INI instance to share the same application.

/D <output> specifies the files for stdout (*.out) and stderr (*.err) messages. Further examples of using the /D option are defined in Table 8-1 on page 66.

DESCRIPTION

The INI executable performs most of the major functions of the data transfer portion of INI product.

This executable is used to:

- Read a map file and indicate if it is syntactically correct.
- Read the map file and create the appropriate records in the database.
- Register the objects with CSA, create the specified Application in the Shared Memory of the local Object Manager, and notify the other Object Managers in the system of the new Top Level Name (application).
- Initiate data transfer (mapping) for one (**map**) or more (**mmap**) applications.
- Re-create the Application to adjust the in memory representation of the application to meet any changes made to the database.
- Remap data to reflect the data transfer changes specified in the database.
- Record changes to the value of specified attributes to the database so that the new values is used the next time the attributes are created.
- Turn on debug printing.
- Stop data transfer (**unmap**).
- Delete the Application from the local OM's Shared Memory and remove it from CSA

The INI executable assumes that the file is in the current directory by default. If the file cannot be found, the INI executable does not start and a message written to the error output.

The **-AN** option can be used to override the default location of the **an_init.cfg** by directly and fully specifying the path name of the **an_init.cfg** file.

The defaults for the various options are:

-SR: 1s for scan rate
-BT: **/opt/aos/data/defaultBTRANS** for Bit Translation Table
-CF: No checkpointing
-AN: **./an_init.cfg**
-WDT: Twice the scan rate. The WDT value may be set higher than this, but not less.
-RD: 120 seconds
-NOP is not taken; **P** mappings are initialized.

For mapping operations, the **<scan rate>** sets the mapping scan rate in seconds. The main event loop of the INI package will run that often. The INI software configures the FoxAPI/AIM*API read scan rate (rsr) is 1/2 this value. The FoxAPI/AIM*API write scan rate is 0.5 s more than the rsr.

For most efficient operation, the **fastest_rsr** attribute in the FoxAPI/AIM*API configuration file (**d:\opt\fox\ais\bin\foxapi.cfg or d:\opt\fox\aim\bin\aimapi.cfg**) should be set to 1/2 the expected INI software scan rate. However, other applications may require faster settings.

Message Forwarding

iaToQ

Receives message and places them in a queue for transmittal over a TCP/IP network.

SYNOPSIS

```
iaToQ <IAName> <serverName> [ -f <fileOfNames>] [ -q <queueSize>]
     [ -sn <serviceName>] [ -h[elp] ]
```

where:

<IAName> - Remote Station's logical/device name. This name must be unique on the Remote Station's network. It may contain A-Z, 0-9, and _ and it must be 12 characters or less.

<serverName> - Network name of the station that is running the **netToIA** process.

<fileOfNames> - File containing up to 20 local device names which define the targeted devices. This file must reside in on both the Remote Station and Local Stations. The default file name is **D:\opt\aos\MsgFwd\data\FileOfNames**. If the pathname supplied is relative the default directory will be used. If the pathname is absolute, the file may be in a different directory.

<queueSize> - Size of message queue used to buffer messages before they are forwarded to the Local Station. The default value is 1000 queue entries.

<serviceName> - Name of the TCP/IP service being used. Default name, **IAMsgFwd**, may not be valid. Be sure that **C:\windows\system32\drivers\etc\services** file has been edited correctly for the selected name.

DESCRIPTION

The process **iaToQ** exists on the Remote Station. This process first creates a "message queue", a shared memory segment of size **<queueSize>**, and creates and initializes semaphores to control access to this queue.

The process then calls the **qToNet** process, which handles communications across the TCP/IP connection, with the Local Station. The process then reads the **<FileOfNames>** file, and registers to use IPC for connectionless data transfer services for all of its aliases.

The **iaToQ** process then enters an infinite loop in which it requests data (connectionless) from each alias, and waits for a reply from any/all of the aliases. When messages are received, they are decoded, split into individual messages, re-formatted, and inserted onto the "message queue".

qToNet

Takes messages from the input queue and places them on the network

SYNOPSIS

```
qToNet <IAName> <serverName> [ -sn <serviceName>] [ -h[elp] ]
```

where:

<IAName> - Logical/device name of the record forwarding program on this system It may contain A-Z, 0-9, and _ and it must be 12 characters or less. In normal operation, this name will be supplied by **iaToQ. qToNet** will append **_Q** to it to create its own name.

<serverName> - Network name of the Remote Station that is running the **netToIA** program.

<serviceName> - Name of the TCP/IP service being used. Default name, **IAMsgFwd**, may not be valid. Be sure that **C:\windows\system32\drivers\etc\services** file has been set up for this name.

DESCRIPTION

The process **qToNet** first creates a socket to the Local Station and then connects to that station.

It then attaches to the existing shared memory segment that will function as the message queue between **qToNet** and **iaToQ**.

It then enters an infinite loop in which it waits for information to be inserted by **iaToQ** onto the message queue. When that event occurs, it retrieves the information from the queue and sends it to the service **netToIA** on the Local Station that is responsible for forwarding the message to the system.

— NOTE

The **qToNet** process is started automatically by the process **iaToQ**. It should not be started manually.

netToIA

Takes messages from TCP/IP network and places them on the control network.

SYNOPSIS

`netToIA [-f <fileOfNames>] [-sn <serviceName>] [-n <altName> ] [-h[elp]]`
where:

<fileOfNames> - File containing up to 20 local device names, each of which has up to targeted remote devices. File must reside in data directory. The default name is **FileOfNames**.

<serviceName> - Name of the TCP/IP service being used. Default name, IAMsgFwd, may not be valid. Be sure that **C:\windows\system32\drivers\etc\services** file has been set up for this name.

<altName> - is the alternate name of the **netToIA** process. It may contain A-Z, 0-9, and _ and it must be 14 characters or less. This option is required only if more than 1 copy of **netToIA** is run in the system. Any unique process name will suffice. The default is **netToIA**.

DESCRIPTION

The process **netToIA** first creates a socket to the Remote Station.

It then waits for Remote Station messages over the socket connection, and when message are received, establishes a link with the Remote Station and forks a child process to handle TCP/IP communications.

This child process reads the **<fileOfNames>** file, and registers to use IPC for connectionless data transfer services for all of its aliases.

It then enters an infinite loop in which it kills any zombie processes, reads the message from the Remote Station, decodes it, and forwards the messages on to the appropriate destination.

If trying to run more than one MsgFwd Remote Station/Local Station pair, where BOTH Local Stations are on the same control network, the **-n <altName>** flag must be provided for one or both of the Local Stations. This is because the Local Station process will register for inter-process communication with control network, and their names must be unique.

AppAlm

Generates a test message for use with Message Forwarding.

SYNOPSIS

```
AppAlm <logicalName> (<stationLBug> | ") (A | K | N) <desc> <alm msg>
<priority> <USEBLK | NOTUSEBLK> <compound> <block>
```

where:

<logicalName> - One of the group names created in the **FileOfNames** file on the Remote Station. This parameter is required. This name could be logical names for stations and supplied programs, e.g. the Historian and the Display Managers are assigned using the System Configurator. The **logicalName** must be no more than 6 characters and is limited to the set of characters expected by the Object Manager for alarm device logical names: A-Z, underscore (_), and 0-9.

<stationLBug> - The letterbug of the Remote Station. This parameter is only required if the target logical name is not unique within the system. The tasks that implement message pass-through in the Gateways are not uniquely named so they would require this parameter.

(**A | K | N**) - **A** for Into Alarm; **K** for Acknowledged; **N** for Return to Normal.

<desc> - The alarm description. This text (32 chars max) will appear on the first line of the CAD entry. This is equivalent to the block description field.

<alm msg> - The message text for the alarm. This text (32 chars max) will appear on the second line of the CAD entry.

<priority> - The Priority of the alarm (1 to 5).

<USEBLK | NOTUSEBLK> - Flag that specifies whether or not to set/reset a specified block name.

<compound> - The compound name of the CIN block to acknowledge.

<block> - The block name of the CIN block to acknowledge.

DESCRIPTION

AppAlm sends the indicated message to the target Alarm Annunciator.

Message Relay

msgRelay

Implements two-way IPC based communications over a customer supplied network.

SYNOPSIS

```
msgRelay <IAName> <serverName> [ -p <fileOfProxies>] [ -q <queueSize>]
[ -sn <serviceName>] [ -h[elp] ]
```

where:

<IAName> - Remote Station's logical/device name. This name must be unique on the Remote Station's network and it must be 12 characters or less.

<serverName> - Network name of the station that is running the matching **msgRelay** process, i.e. a msgRelay process using the same service name.

<fileOfProxies> - File containing up to 20 local logical names which define the targeted programs on the Remote Station. The default file name is **D:\opt\aos\MsgRelay\data\FileOfProxies**. However, the -p option allows a full pathname to be specified which allows the file to be placed in any directory.

<queueSize> - Size of message queue used to buffer messages before they are forwarded to the Local Station. The default value is 1000 queue entries.

<serviceName> - Name of the TCP/IP service being used. Default name, **IAMsgRelay**, may not be valid. Be sure that **C:\windows\system32\drivers\etc\services** file has been edited correctly for the selected name.

DESCRIPTION

The **msgRelay** program runs in pairs. The local and remote copies must both reference the same **serviceName**. By convention, the **IAName** for each member of the pair should be the same on each machine.

Each instance of **msgRelay** is limited to representing 20 processes on the Remote Station. If more than 20 processes need to be represented, multiple copies of **msgRelay** will be required.

The default configuration files are expected in **D:\opt\aos\MsgRelay\data**, but this can be over ridden by the **-p** option.

EXAMPLE

This example shows a default startup on the Local Station:

```
D:/opt/aos/MsgRelay/exe/msgRelay MSGRELAY nisaw2 &
```

This example shows the corresponding startup on the Remote Station. It specifies a proxy file called FileOfTargets that should be in **D:\opt\aos\MsgRelay\data**.

```
D:/opt/aos/MsgRelay/exe/msgRelay MSGRELAY nisaw3 -p FileOfTargets &
```

Map File Building

mkBlkIndex

Reads the workfiles and creates a block index file.

SYNOPSIS

```
mkBlkIndex <wfName> [...<wfName>] | sort -o /opt/aos/wf/BlockIndex
```

where:

<wfName> - is a workfile. Generally, it is one generated by **GatherWorkFiles**.

DESCRIPTION

The script **GatherWorkFiles** copies the workfile directory for a specified set of Control Stations to **D:\opt\aos\wf**. Once collected, these workfiles can be used to build the index of blocks for the system using **mkBlkIndex**. The index can be used by **mkINIMapFile** to create map files.

EXAMPLE

Typical usage is as follows:

```
cd D:/opt/aos/scripts
./GatherWorkFiles
../exe/mkBlkIndex ../wf/*/*.wf | sort -o ../wf/BlockIndex
../exe/mkINIMapFile ../wf/BlockIndex -d ../tmp <../data/mapDef
```

mkINIMapFile

Uses a block index and a set of rules to create **.map** files.

SYNOPSIS

```
mkINIMapFile <blkIndexFile> -d <mapFileDir> -INI15 <prefix> <mapFileRules>
```

DESCRIPTION

The script **GatherWorkFiles** copies the workfile directory for a specified set of Control Stations to **D:\opt\aos\wf**. Once collected, these workfiles can be used to build the index of blocks for the system using **mkBlkIndex**. The index can be used by **mkINIMapFile** to create **.map** files.

The arguments to **mkINIMapFile** are:

<blkIndexFile> - which is the path name to a block index created by **mkBlkIndex**. By convention, the name is **D:/opt/aos/wf/BlockIndex**.

-d <mapFileDir> - specifies the directory that is to hold the **.map** files. Generally, this is a user created directory.

-INI15 <prefix> - specifies the prefix to be prepended to the remote compound name to create the local application name.

<mapFileRules> - specifies the Map Definition file.

mkINIMapFile reads the Map Definition from the standard input so connecting the standard input of the program to a Map Definition file specifies the rules that **mkINIMapFile** is to use.

If the mapping operator specified in the Map Definition file is TT_x where x is from 1 to 8, **mkINIMapFile** will generate records for the required triggers. These records will have constant mappings. If the attribute being mapped is a Sequence Block string parameter (**SN0001** through **SN0010**), the trigger will be read mapped to the **MSGNO** parameter of the sequence block. The **MSGNO** parameter is a long integer that is incremented with wrap-around each time a string parameter is changed by the Sequence Block.

If the map file is being built to replace an INI15, the **-INI15** option must be used. The INI15 required the local compound to have a two-character prefix to the compound name in the remote system. The **-INI15** option allows the user to specify this prefix.

EXAMPLE

Typical usage for new installations is as follows:

```
cd /opt/aos/scripts
./GatherWorkFiles
../exe/mkBlkIndex ../wf/*/*.wf | sort -o ../wf/BlockIndex
../exe/mkINIMapFile ../wf/BlockIndex -d ../tmp <../data/mapDef
```

For example, Map Definition File entries like these:

```
# IND Block parameters
IND   SN0001   S     80 TT_1  4
IND   SN0002   S     80 TT_2  4
IND   SN0003   S     80 TT_3  4
IND   SN0004   S     80 TT_4  4
IND   SN0005   S     80 TT_5  4
IND   SN0006   S     80 TT_6  4
IND   SN0007   S     80 TT_7  4
IND   SN0008   S     80 TT_8  4
IND   SN0009   S     80 TT_1  4
IND   SN0010   S     80 TT_2  4
IND   MSGNO    L     .  R     6
```

will produce map file records like these:

```
DMCPLUSDOP:COUNTDOWN.MSGNO @ R 6 0 0.500000
DMCPLUSDOP:COUNTDOWN.SN0001 @ TT_1 4 "" 80
DMCPLUSDOP:COUNTDOWN.SN0002 @ TT_2 4 "" 80
DMCPLUSDOP:COUNTDOWN.SN0003 @ TT_3 4 "" 80
DMCPLUSDOP:COUNTDOWN.SN0004 @ TT_4 4 "" 80
DMCPLUSDOP:COUNTDOWN.SN0005 @ TT_5 4 "" 80
DMCPLUSDOP:COUNTDOWN.SN0006 @ TT_6 4 "" 80
DMCPLUSDOP:COUNTDOWN.SN0007 @ TT_7 4 "" 80
DMCPLUSDOP:COUNTDOWN.SN0008 @ TT_8 4 "" 80
DMCPLUSDOP:COUNTDOWN.SN0009 @ TT_1 4 "" 80
DMCPLUSDOP:COUNTDOWN.SN0010 @ TT_2 4 "" 80
DMCPLUSDOP:COUNTDOWN.trgr_1 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
DMCPLUSDOP:COUNTDOWN.trgr_2 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
DMCPLUSDOP:COUNTDOWN.trgr_3 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
DMCPLUSDOP:COUNTDOWN.trgr_4 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
DMCPLUSDOP:COUNTDOWN.trgr_5 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
DMCPLUSDOP:COUNTDOWN.trgr_6 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
DMCPLUSDOP:COUNTDOWN.trgr_7 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
DMCPLUSDOP:COUNTDOWN.trgr_8 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
```

Typical usage for an INI15 upgrade is as follows:

```
cd D:/opt/aos/scripts
./GatherWorkFiles
../exe/mkBlkIndex ../wf/*/*.wf | sort -o ../wf/BlockIndex
../exe/mkINIMapFile ../wf/BlockIndex -INI15 I2 -d ../tmp
<../data/mapDef
```

For example, Map Definition File entries like these:

```
# IND Block parameters
IND   SN0001   S     80 TT_1  4
IND   SN0002   S     80 TT_2  4
IND   SN0003   S     80 TT_3  4
IND   SN0004   S     80 TT_4  4
IND   SN0005   S     80 TT_5  4
IND   SN0006   S     80 TT_6  4
IND   SN0007   S     80 TT_7  4
IND   SN0008   S     80 TT_8  4
IND   SN0009   S     80 TT_1  4
IND   SN0010   S     80 TT_2  4
IND   MSGNO    L     .  R     6
```

will produce map file records like these:

```
I2DMCPLUSDOP:COUNTDOWN.MSGNO DMCPLUSOP:COUNTDOWN.MSGNO R 6 0 0.500000
I2DMCPLUSDOP:COUNTDOWN.SN0001 DMCPLUSOP:COUNTDOWN.SN0001 TT_1 4 "" 80
I2DMCPLUSDOP:COUNTDOWN.SN0002 DMCPLUSOP:COUNTDOWN.SN0002 TT_2 4 "" 80
I2DMCPLUSDOP:COUNTDOWN.SN0003 DMCPLUSOP:COUNTDOWN.SN0003 TT_3 4 "" 80
I2DMCPLUSDOP:COUNTDOWN.SN0004 DMCPLUSOP:COUNTDOWN.SN0004 TT_4 4 "" 80
I2DMCPLUSDOP:COUNTDOWN.SN0005 DMCPLUSOP:COUNTDOWN.SN0005 TT_5 4 "" 80
I2DMCPLUSDOP:COUNTDOWN.SN0006 DMCPLUSOP:COUNTDOWN.SN0006 TT_6 4 "" 80
I2DMCPLUSDOP:COUNTDOWN.SN0007 DMCPLUSOP:COUNTDOWN.SN0007 TT_7 4 "" 80
I2DMCPLUSDOP:COUNTDOWN.SN0008 DMCPLUSOP:COUNTDOWN.SN0008 TT_8 4 "" 80
I2DMCPLUSDOP:COUNTDOWN.SN0009 DMCPLUSOP:COUNTDOWN.SN0009 TT_1 4 "" 80
I2DMCPLUSDOP:COUNTDOWN.SN0010 DMCPLUSOP:COUNTDOWN.SN0010 TT_2 4 "" 80
I2DMCPLUSDOP:COUNTDOWN.trgr_1 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
I2DMCPLUSDOP:COUNTDOWN.trgr_2 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
I2DMCPLUSDOP:COUNTDOWN.trgr_3 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
I2DMCPLUSDOP:COUNTDOWN.trgr_4 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
I2DMCPLUSDOP:COUNTDOWN.trgr_5 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
I2DMCPLUSDOP:COUNTDOWN.trgr_6 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
I2DMCPLUSDOP:COUNTDOWN.trgr_7 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
I2DMCPLUSDOP:COUNTDOWN.trgr_8 DMCPLUSDOP:COUNTDOWN.MSGNO R 5 0 0.5
```

NOTE

The local name and the remote name are identical except the local name has the specified prefix.

File Formats

FileOfNames

The file **FileOfNames** is the Message Forwarding component of the INI package data file, used on both the Remote Station and Local Stations. It defines the ultimate destination of any messages forwarded from the Remote Station.

The **FileOfNames** contains two types of records: Comment Records and Destination Records.

There are two types of Comment records:

- Those that have a # in the first column followed by any desired text.
- Those that contain only white space (blanks and tabs).

There are three types of Destination Records:

- Alarm Device specification records.
- Alarm Contact specification records.
- Program specification records.

If a message is to be forwarded to both a list of Alarm Devices and a list of Alarm Contacts, the Device and Contact Records should be adjacent to each other in the file with the Device record first.

The format for Destination Records contains three fields: **GroupName**, **Type**, and **DeviceList**.

1. The **GroupName** field specifies the name used on the Remote Station as a target for messages.
2. The **Type** field specifies the type of device to which the message should be forwarded. The supported device types are as follows: Horns, Alarm Devices, e.g., printers and stations, and application programs that use connectionless IPC.
3. The **DeviceList** field which is a list of Object Manager Pathnames or logical names to which the message should be forwarded. Blanks or tabs should separate the names in the Device List. Horn names should be Boolean Object Manager variables, e.g., **C.P.COUT**. The logical names should be the logical name of the target program, e.g., **LP00** for a printer, **AW0001** for a station, or some program.

— NOTE

Do not put line breaks in these records, if the number of names required does not seem to fit just keep typing and use the wraparound feature of vi or textedit to extend the line.

An example file is found in the data directory (**D:\opt\aos\MsgFwd\data**).

To be more specific,

1. The file FileOfNames defines up to twenty (20) logical names on the Remote Station which represent the devices and contacts on the local system to receive the messages.
2. The file FileOfNames defines the devices, programs and contacts in the Local Station that should receive alarm message from the Remote Station.
3. The file FileOfNames may specify as many as twenty (20) different GroupNames. At least one must be specified.

4. The file FileOfNames contains two types of records: Comments and Destinations.
5. The Comment Records have a hash mark (#) in the first column and arbitrary text thereafter or they may be completely blank.
6. The Alarm Device Record format is as follows:

   <groupName> <RD | LD> <DeviceName> [<DeviceName> ...]

   where:

   <groupName> - is the group name or alias for the list of devices that are to receive the forwarded messages. The **groupName** must be no more than 6 characters and is limited to the set of characters expected by the Object Manager for alarm device logical names: A-Z, underscore (_), and 0-9.

   RD | LD> - is **RD** for a remote device is **LD** for a local device (** **LD** support is currently not implemented).

   <DeviceName> - is the logical name for the device (printer/station/Historian) to which the message is to be forwarded. No more than twenty (20) names may be specified.

Each group name specified may have only one Device Record. The Contact and Device records that share a group name must be consecutively placed in the file and the Device record must come first. No more than twenty unique (20) target Device Names may be specified. Extra names will be ignored.

7. The Contact Record format is as follows:

   <groupName> <RH | LH> <ContactNames>

where:

<groupName> - is the group name or alias for the list of devices that is to receive the forwarded messages. The groupName must be no more than 6 characters and is limited to the set of characters expected by the Object Manager for alarm device logical names: A-Z, underscore (_), and 0-9.

<RH | LH> - is **RH** for a remote contact name (horn) is **LH** for a local contact name (horn) (** **LH** support is currently not implemented).

<ContactNames> - are the names of the CBPs in the Local Station that should be closed for each priority. The format is:

[<PriorityOneContact> | NONE]

[...[<PriorityTwoContact> | NONE]

[...[<PriorityThreeContact> | NONE]

[...[<PriorityFourContact> | NONE]

[...[<PriorityFiveContact> | NONE]

[...[<PrioritySystemContact> | NONE]]]]]]

Each group name specified may have only one Contact Record. The Contact and Device records that share a group name must be consecutively placed in the file and the Device record must come first.

8. The Program Record format is as follows:

   <groupName> <RP | LP> <DeviceName> [**<DeviceName> ...**]

   where:

   <groupName> - is the group name or alias for the list of programs that are to receive the forwarded messages. The **groupName** must be no more than 6 characters and is limited to the set of characters expected by the Object Manager for alarm device logical names: A-Z, underscore (_), and 0-9.

   <RP | LP> - is **RP** for a remote Device is **LP** for a local Device (** **LP** support is currently not implemented).

   <DeviceName> - is the logical name of the target program on the Local Station.

   Each group name specified may have only one Device Record. The Contact and Device records that share a group name must be consecutively placed in the file and the Device record must come first. There may be no more than twenty (20) names. Extra names will be ignored.

The following table presents several examples and an explanation of when each could be used:

Table 8-4. Examples of FileOfNames Configurations

Example	Explanation of Use

`ALMTGT RD WP0001 LP01`
`ALMTGT RH HORN:1.IN HORN:2.IN`
`NONE NONE NONE NONE`
	This combination is typical of the setup required for Process Alarm Forwarding.

`SYSTGT RD LP01`
`SYSTGT RH NONE NONE NONE NONE`
`NONE HORN:S.IN`
	This combination is appropriate for System Messages.

`PGMTGT RP AMI101`	This combination is appropriate for OAJ messages.
`PGMTGT RP MY_PGM`	This combination is appropriate for a program that receives connectionless IPC messages.

FileOfProxies/FileOfTargets

The file **FileOfProxies** is the portion of the Message Relay component of the INI package data file used on the local station to define the servers in the Remote Station that are to be made available locally.

The file **FileOfTargets** is the complement of the **FileOfProxies** and it is used on the Remote Station to identify the programs on the Local Station.

The file formats are identical.

The **FileOfProxies** contains two types of records: Comment Records and Proxy Records.

There are two types of Comment records:

- Those that have a # in the first column followed by any desired text.
- Those that contain only white space (blanks and tabs).

There is only one type of Destination Record. The format of the Destination Record is:

<LogicalName> [**SEND|BCAST**]

The second token on the line is optional. A connectionless point to point relay is assumed by default.

Map Definition File (mapDef)

The **mapDef** file is used by the program **mkINIMapFile** to define the CBPs from the Remote Station that are to be represented in the Local Station. See page 31 and page 83 for details on its use.

The file contains records defining the points to include in the INI's map file. Each record specifies a block type and parameter to be recorded. Six fields are used. Two fields are required. Fields three, four, five, and six are optional.

The format of the records in the map definition file is:

```
blkTypeName pName [rName [[delta_in_%_of_range | strlen] [mapOper
[dataType]]]
```

A dot (.) can be used to specify the use of default value. The following table defines each field and the default for each field:

Table 8-5. Possible Values of Fields in the Map Definition File

Field Name	Default	Possible Values
rName	RI1	R[I|O][1|2|3] for most floating point variables B for Boolean, Packed Boolean, and Packed Long I for Integer and Long S for String
delta_in_%_of_range or strlen	1.0 32	Floats: 0.0 < delta <= 100.0; Non-string data types: 0.5 String: Length of String
mapOper	R	See the table on page 95 for the complete list of mapping types.
datatype	3	1 - CHARACTER; Character 2 - INTEGER; 16 bit integer 3 - FLOAT; 32 bit float 4 - STRING 5 - OM_BOOL; Boolean 6 - OM_LNG_INT; 32 bit integer 8 - CIO_SHORT; 8 bit integer 9 - OM_S_PKBOL; 16 bit packed Boolean 10 - OM_L_PKBOL; 32 bit packed Boolean

The Map Definition File can be used for attributes that are not actively mapped. The C (constant) and **U** (updated/checkpointed) mapping operators allow for the definition of a parameter that is not attached to the remote block. The initial value for these parameters is the value from the work file used in conjunction with the Map Definition File when **mkINIMapFile** is run. Because reasonable initial values are available from the work file and because these values seldom change, it is reasonable to "map" parameters like **DESCRP**, **EO1**, and **EI1** as constants.

The FoxView display uses the block parameter **Rxy** (where x=**I** or O and y=**1**, **2**, or **3**) to support the default displays and other display items that require the high scale, low scale, and change delta. Since the INI package automatically creates the **Rxy** parameter, **Rxy** should never by included explicitly in the Map Definition File.

The INI package creates the Rxy parameter automatically when the **HSCxy** parameter is encountered.

If the **HSCxy** parameter is specified, all three parameters of the triplet used to support the Rxy parameter must be specified. To be specific, if **HSCO1** is specified, **LSCO1** and **DELTO1** must also be specified. Likewise, if **HSCI3** is defined so too must **LSCO1** and **DELTO1**.

See the example file below for the examples on how to use of **HSCxy**, **DESCRP**, **EO1**, and other parameters.

EXAMPLE

```
# AIN  Block    parameters
AIN    DESCRP   S     32    C    4
AIN    DESCRP   S     32    C    4
AIN    PNT      RO1    1    R    3
AIN    HSC01    RO1   32    C    3
AIN    LSC01    RO1   32    C    3
AIN    DELT01   RO1   32    C    3
AIN    EO1      S     32    C    4

# AOUT Block    parameters
AOUT   DESCRP   S     32    C    4
AOUT   OUT      RO1    5    R    3
AOUT   HSC01    RO1   32    C    3
AOUT   LSC01    RO1   32    C    3
AOUT   DELT01   RO1   32    C    3
AOUT   EO1      S      1    C    4

# PID  Block    parameters
PID    DESCRP   S     32    C    4
PID    MEAS     RI1    1    R    3
PID    SPT      RI1    1    P    3
PID    HSCI1    RI1   32    C    3
PID    LSCI1    RI1   32    C    3
PID    DELTI1   RI1   32    C    3
PID    EI1      S     32    C    4

PID    OUT      RO1    5    P    3
PID    HSC01    RO1   32    C    3
PID    LSC01    RO1   32    C    3
PID    DELT01   RO1   32    C    3
PID    EO1      S     32    C    4

PID    LR       B      .    P    5
PID    MA       B      .    P    5
PID    BLKSTA   B      .    R    10
PID    ALMSTA   B      .    R    10
```

The Map File

User supplied map (**.map**) files provide project-specific information to INI and AOS packages. The following sections discuss the format of this file.

The map file enumerates all of the objects and attributes for a given application and defines how each attribute is handled. There are two types of lines in a map file: *Object Definition Lines* and *Attribute Definition Lines*. These lines are explained in greater detail below.

Object Definition Lines

Object Definition Lines are relatively simple. An Object Definition Line consists of two elements: the object name and its type. An example of an Object Definition Line could be:

```
OHTEMP     PIDA
```

where **OHTEMP** is the name of the object, and **PIDA** is the object type.

Attribute Definition Lines

Attribute Definition Lines are much more complicated. These lines define the attributes found within an object and have the following general format:

```
<AttributeName> <UnderlyingParameter> <MT> <DT> <IV> <CD/SL>
```

where:

<AttributeName> - The name of the local AOA. It is mapped to a remote tag.

<UnderlyingParameter> - The name of the tag in the Remote Station to which the **AttributeName** is mapped.

<MT> - The mapping type, see Table 8-6.

<DT> - The data type for **AttributeName**, see Table 8-6.

<IV> - The initial value for **AttributeName**, see Table 8-6.

<CD/SL> - The change delta or string length for **AttributeName**, see Table 8-6.

There is much variation within this format. The following are examples of valid Attribute Definition Lines:

```
EXAMPL:OHTEMP.APPNAM                          X   C     4     "EXAMPL"
                    12
PRETEND:OBJECT.ATTR CMPD:BLOCK.PARMP          P   3     1       0.001
BITTWIDDLE:HH.CIN BITTWIDDLE:ALMSTA.ALMSTA.HH R   5     1       0.001
```

The individual fields on the Attribute Definition Line can be further explained as follows:

Table 8-6. Fields in an Attribute Definition Line

Field Name	Description
`Attribute Name`	Fully specified name of the AOA in AOA format. The application name is optional.
`Underlying Parameter`	Fully specified name of the AOA, CBP or Shared Variable, or X if not used. If in INI mode, there are two other interpretations of this field: ♦ If the field contains an @ symbol, the Attribute Name is linked to a target in the Remote Station with the same name as the Attribute Name. The use of this symbol makes reading AOS map files simpler. ♦ If the mapping is an **I** mapping, the **INI** connects the Attribute Name to the Underlying Parameter using the ICC driver task as with AOS mode. However, the INI software also connects the Attribute Name to a remote tag of the same name as if the mapping were an **R** mapping. This makes creating peer-to-peer connections between systems simpler.
`Mapping Type (MT)`	Mapping Options. See Table 8-7 for a full list and a description of each.
`Data Type (DT)`	The data type for the attribute: **1 - CHARACTER 5 - OM_BOOL 9 - OM_S_PKBOL** **2 - INTEGER 6 - OM_LNG_INT 10 - OM_L_PKBOL** **3 - FLOAT 7 - unused** **4 - STRING 8 - OM_SHORT_INT**
`Initial Value (V)`	Initial Value for the Attribute during its creation. If the attribute is a checkpointed attribute, the initial value is used for the first creation. Thereafter, the checkpointed value is used during creation.
`Change Delta/String Length (SD/SL)`	Change Delta or the length of the string if the data type is **STRING**

NOTE

1. Each line of the mapping file represents either an AOA or an AO definition. Blank lines are allowed and any line starting with a # symbol is a comment.

2. Attributes of Applications are supported through the use of a special object definition line. This object definition line has a single word (**COMPND**) on it. The lines that follow must have an empty string for the object name, i.e., either**:** **.PERIOD** or **<appName>:.PERIOD**.

3. An object used in an AOA definition must be previously defined on an earlier line in the mapping file.

4. Each field on the line is separated by any amount of white space. Each AOA definition line must contain all 6 fields. Dummy CBP names, e.g., 'X', are entered for U and C mappings in the 'Underlying Parameter' field.

5. The initial values of a character data type attribute must be a non-space, printable ASCII character.

6. String values are enclosed by double quotes. The length of a string is specified in its deadband field. The minimum length of a string is 32 characters; any smaller value is ignored. The maximum length is 254.

7. There are no restrictions on the name or path of the mapping file.

8. The application part of an AOA name in the Attribute Name field of a map definition line is ignored if supplied.

9. The TYPE attribute of an object is automatically created and initialized by AOS when it processes an AO definition line in the mapping text file. Thus, there should be no TYPE AOA definitions in the mapping text file.

10. Do not create attributes with names with the form **R**[**I**|**O**][**number**], e.g., **RI1** or **RO1**. These names are reserved and automatically created when an attribute whose name is **HSC**[**I**|**O**][**number**] is encountered.

Mapping Type Explained

The Attribute Line Definition field, **<MT>** or Mapping Type, determines the direction of data flow between the **AttributeName** and the **UnderlyingParameter**, as well as the method and frequency with which the data is transferred. In the following example the Mapping Type is **P**, a Put mapping.

```
<AttributeName>       <UnderlyingParameter>  <MT>  <DT>  <IV>  <CD/SL>
PRETEND:OBJECT.ATTR   CMPD:BLOCK:PARM          P     3      1     0.001
```

A summary of these options and their meaning follows below:

Table 8-7. Mapping Type Summary

Operator	Definition	Requires Mapping	Data Types	Type Casting	Bit Extraction
A	Specifies the start of an Alarm Object.	Yes	STRING	N/A	N/A
C	Constant value. This value is not changed by Application Object Services. However, it is initialized to a specific value when the object is created.	No	All	N/A	N/A
G	Uses a connectionless get on the value when the object is created. It is never re-read. If the connectionless get fails, the value in the database is use.	No	All	No	No
g	Uses a connectionless get on the value when the object is created. It is re-read when the AOS is signaled to do so. The trigger is the **.g_trgr** parameter of the object.	Yes	All	No	No
I	Transfers data from the AOA to the target CBP using a read list opened by the C:B. AOS configures the name of the AOA into the CP's Control Block using the ICC Driver Task. These connections are identical to connections between two blocks. Status bits are transferred as well.	No	All except STRING	N/A	N/A

Table 8-7. Mapping Type Summary (Continued)

Operator	Definition	Requires Mapping	Data Types	Type Casting	Bit Extraction
p	Lower-case P This mapping is used to allow a workstation behind a firewall to push data from its protected system to an unprotected workstation outside the firewall. This is the core technology behind the *Isolation Station*. In this usage, the same .map file is used for aos and an INI instance. The -MapAlias option of the aos instance is used to allow both instance to run and share the application. For STRING variables: "p" Mappings * aos will map "p" as "g" * INI will map "p" as "P" For non-STRING variables: "p" Mappings * aos will map "p" as "R" * INI will map "p" as "P" String variables will move from the protected to unprotected system on a change driven basis. AOS will move them from the CP to the INI host when g_trgr toggles. Non-String variables will move from the protected system to the unprotected system on a change driven basis through the INI. The will move from the CP to the INI host on a change driven basis using AOS.	Yes	All	No	No
P	Upper-case P Uses a Change Driven connection from an AOA to obtain values to be stored in the specified target location, typically a CBP or SV. The value is written to the target using connectionless transfers so the target is not secured to this application object. The ACK, BAD, and OOS status bits are transferred as well.	Yes	All	No	No
R	Uses a Change Driven connection from OM vales, typically CBPs or SVs, to obtain values to be stored in the specified AOA. Status bits are transferred as well.	Yes	All except STRING	Yes	Yes
TT_x	Triggered read. The triggers are the **.trgr_x** attributes for the same object. There are 8 triggers per object: the attributes **.trgr_1** through **trgr_8**.	Yes	All	Yes except STRING	Yes except STRING

Table 8-7. Mapping Type Summary (Continued)

Operator	Definition	Requires Mapping	Data Types	Type Casting	Bit Extraction
U	Checkpoints the AOA value to the database on either a periodic or triggered basis.	Yes	All	N/A	N/A
W	Lower-case W This mapping is used to allow a workstation behind a firewall to push data from its protected system to an unprotected workstation outside the firewall. This is the core technology behind the *Isolation Station*. In this usage, the same .map file is used for aos and an INI instance. The -MapAlias option of the aos instance is used to allow both instances to run and share the application. For STRING variables: "w" Mappings * aos will map "w" as "g" * INI will map "w" as "W" For non_STRING variables "w" Mappings * aos will mp "w" as "R" * INI will map "w" as "W" String variables will move from the protected to unprotected system on a change driven basis. AOS will move them from the CP to the INI host when g_trgr toggles. Non-String variables will move from the protected system to the unprotected system on a change driven basis through the INI. They will move from the Cp to the INI host on a change driven basis using AOS.	Yes	All except STRING	No	No
W	Upper-case W Uses a Change Driven connection from an AOA to obtain values to be stored in the specified target location, typically a CBP or SV. The value is written to the target using connection-based transfers. Therefore, the target is secured to this application object. Status bits are not transferred since the FoxAPI/AIM*API **wrtval** call is used.	Yes	All except STRING	No	No

NOTE

In AOS mode: ICC (I) mappings are supervised within a CP. When mapping is turned on, AOS optionally informs the ICC to connect an AOA to a CBP. When mapping is turned off, AOS optionally informs the ICC to disconnect the AOA from the CBP indicating that the connection is no longer in use. ICC mapping is the best way to move data into the CP. However, it does require that the target parameter support a connection and not all parameters support input connections. In most other respects, I mappings are treated as constants, C mappings.

In INI mode: ICC (I) mappings work as described above. However, they are also treated as R mappings, i.e., the data for the local AOA is assumed to come from a remote tag whose name is the same as the name of the AOA.

For both modes: For W, P, and I mappings, the AOA source value is initialized to its associated CBP sink value when mapping is turned on. This "backwards" copy is done to reduce the possibility of process upsets when mapping is turned on. It is possible to disable the readback for P mappings since some targets, e.g., the Spectrum Master Gateway, do not allow read backs for some targeted parameters, e.g., CCSPT.

Type Casting Explained

The **R**, i.e., read, mapping supports mapping from one data type to another. Type casting is supported from all OM Scalar data types to all OM Scalar data types. The scalar data types are given in the following table along with the values to the mapped value is clamped before it is stored into the AOA.

Table 8-8. Clamp Limits on Data Type Casting

Target Data Type	Lower Limit	Upper Limit
Character	-128	127
Boolean	0	1
Short Integer (8 bits)	-128	127
Float (32 bits)	(float) 1.40129846432481707e-45	(float) 3.40282346638528860e+38
Packed Boolean (16 bits)	0	65 535
Packed Long (32 bits)	0	4 294 967 295
Integer (16 bits)	-32 768	32 767
Long Integer	-2 147 483 648	2 147 483 647

Bit Manipulation Explained

The bit manipulation syntax is supported only in **R**, i.e., read, mappings. This syntax allows the user to extract and negate particular bits from the mapped OM Variable value or status word and store the result in an Application Object Attribute.

Typical mapping file lines are as follows:

Table 8-9. Example Mapping Lines with Bit Manipulations

Mapping Line	Description
`AOA CBP R 5 1.0 0.001`	Standard mapping of a CBP to an AOA.
`AOA C:B.MEASHI.~ R 5 1.0 0.001`	Negates value of Measurement High Indicator (C:B.MEASHI) and stores it in AOA.
`AOA CBPAKCIN.B00 R 5 1.0 0.001`	Extracts B00 from the value of the CBPAKCIN and stores it in AOA.
`AOA C:B.SUPBCO.ACK R 5 1.0 0.001`	Extracts ACK bit from the status word of C:B.SUPBCO and stores it in AOA.
`AOA CBPAKCIN.~B00 R 5 1.0 0.001`	Extracts and negates B00 from value of CBPAKCIN and stores it in AOA.

A set of bit mnemonics, e.g., **ACK**, is supplied with AOS. The file used to define the mnemonics is **D:\opt\aos\data\BTRANS.Vxy** where **x** is the Control Core Services/I/A Series Version Major number, e.g., **8**, and **y** is the version Minor number, e.g., **.5**. Thus the file for I/A Series v8.5 is **D:\opt\aos\data\BTRANS.V85**. The AOS/INI scripts automatically detect the software version and uses the appropriate translation file.

The user may alter these translation files to include project specific mnemonics. However, please note that upgrades to AOS/INI overwrite the user's changes. It is also important to note that Bit Manipulation on a float causes the float value to be taken as 1 if it is non-zero and 0 otherwise. Thus, you cannot extract a particular bit from a Float. The tables on the following pages list the available mnemonics.

Table 8-10. Status Word Bit Mnemonics

Name	Source	Bit #
BAD	S	08
SECURE	S	09
FS	S	09
ACK	S	10
INITC	S	10
OOS	S	11
SHADOW	S	12
LHI	S	13
LLO	S	14
ERROR	S	15
INITU	S	15

Table 8-10. Status Word Bit Mnemonics (Continued)

Name	Source	Bit #
SE	V	29
SC	V	30

Table 8-11. Generic Bit Mnemonics for Value Extraction

Name	Source	Bit #
B00	V	15
B01	V	14
...	...	...
B15	V	00
L00	V	31
L01	V	30
...	...	...
L31	V	00

NOTE

1. S designates that the source of the bit value is the OM Value Record's status word.
2. V designates that bit is taken from OM Value Record's value word.
3. Bits are numbered from 0 to 15 where Bit 15 is the MSB of the standard system 16 bit integer and Bit 00 is the LSB of the same word.

Table 8-12. Generic Bit Mnemonics for Status Extraction

Name	Source	Bit #
00	S	15
01	S	14
02	S	13
03	S	12
04	S	11
05	S	10
06	S	09
07	S	08
08	S	07
09	S	06
10	S	05
11	S	04
12	S	03

Table 8-12. Generic Bit Mnemonics for Status Extraction (Continued)

Name	Source	Bit #
13	S	02
14	S	01
15	S	00

Table 8-13. Alarm Indicator Bits from ALMSTA

Name	Source	Bit #
TRIP	V	31
UNACK	V	30
INH	V	29
OOR	V	28
OPERR	V	27
ST	V	26
HH	V	25
LL	V	24
RT	V	23
BD	V	22
HD	V	21
LD	V	20
HO	V	19
LO	V	18
HA	V	17
LA	V	16

NOTE

1. S designates that the source of the bit value is the OM Value Record's status word.
2. V designates that bit is taken from OM Value Record's value word.
3. Bits are numbered from 0 to 15 where Bit 15 is the MSB of the standard system 16 bit integer and Bit 00 is the LSB of the same word.

*Appendix A. FoxAPI/AIM*API Configuration Files*

*This appendix contains example FoxAPI/AIM*API configuration files.*

FoxAPI/AIM*API

The three FoxAPI/AIM*API configuration files are: **foxapi.cfg, aimapi.cfg** and **an_init.tcp** which reside on the Remote Station and **an_init.cfg** which resides in the data sub-directory of the project directory.

foxapi.cfg/aimapi.cfg - A Remote Station File

— NOTE

For INI applications, the **foxapi.cfg/aimapi.cfg** file on the Remote Station must be modified. The document *FoxAPI Information Bulletin* included in the INI deliverable contains information on the settings found in this file.

```
maxobj      =   40000
maxqo       =   1
maxch       =   1
maxds       =   300
crnpak      =   0
cwnpak      =   1
maxlog      =   1000
ctdlay      =   200
ddctly      =   0
logtrm      =   0
docmsg      =   0
wfilt       =   0
trcast      =   0
trccts      =   0
trcef       =   0
trcfil      =   0
trclok      =   0
trcx25      =   0
trcbrw      =   0
open_wait   =   1
uread_direct  = 0
nocsaonread   = 1
fastest_rsr   = 5
use_omopen    = 1
ia_badstat    = 0
skip_omread   = 0
protect_index = 0
rmtid = 1AW51A 1AW51A / R/W 20 -Series
rmtid = 1AW51B 1AW51B / R/W 20-Series
rmtid = 1AW51C 1AW51C / R/W 20-Series
```

an_init.tcp - A Remote Station File

— NOTE

For INI applications, the **an_init.tcp** file must be created and used on the Remote Station. Follow the procedures in the B0193UC, *I/A Series System FoxAPI Installation Guide*, to create an **an_init.tcp** file for your system.

```
[AISnet]
Protocol        = TCPIP
PrintErr        = 0
TraceLevel      = 0
LogFilePrefix   = ./an
#StderrLog      = stderr No_Timeout
No_Timeout      = 0
Host            = NISAW3
NrServer        = 10 MaxEnt
MaxEnt          = 500
srvrtimeout     = 32000
Machine_Package_Authorization]
HPUX_AISnet     = Not Authorized
VAXVMS_AISnet   = Not Authorized
SUNSPARC_AISnet = 86da29182
DOSPC_DOSDM     = 86da29182
DOSPC_FOXDM     = 86da29182
DOSPC_FoxEXP    = Not Authorized
DOSPC_IASRVR    = Not Authorized
DOSPC_IASRV2    = Not Authorized
DOSPC_ODBC      = Not Authorized
DOSPC_FOXHIS    = Not Authorized
[Machine_Access]
SUNSPARC_001    = 1AW51B
SUNSPARC_002    = 1AW51C
SUNSPARC_003    = 1AW51E
[Package_Access]
AISnet_001      = root
[Security_Access]
root            = "System   |Open Sets|W Files  |W Objects|Read Only
[TCPIP]
*servr          = "127.0.0.1 55555 /dev/tcp 1024"
```

— NOTE

The number associated with **SUNSPARC_AISNET** is the NetFoxAPI/NetAIM*API authorization code. See “NetFoxAPI/ NetAIM*API Licensing” on page 107 for instructions on licensing the NetFoxAPI/NetAIM*API software.

There are times when NetFoxAPI/NetAIM*API tracing is required. To enable this, revise the indicated lines as follows:

```
PrintErr   = 2
TraceLevel  = 1
```

an_init.cfg - A Local Station File

For INI applications, the **an_init.cfg** file must be created on the Remote Station using the **an_setup** program. As part of the use of **an_setup** program, this file is created and saved to a floppy diskette. It is actually used on the Local Station. Follow the procedures in the B0193UC, *I/A Series System FoxAPI Installation Guide*, to create an **an_init.cfg** file for your system.

The **an_init.cfg** file is used by the INI to determine the NetFoxAPI/NetAIM*API Server that it should use. Its location depends on how the INI will be started:

- If the **INI70** executable is being run directly, this file should be in the directory from which the INI is launched.
- If the INI mapping scripts (**INIMAP** and **INIMMAP**) are used to start the **INI70** executable, the **an_init.cfg** file should be in the directory **D:\opt\aos\data** unless the **-M** option is specified. If the **-M** option is used, it must be in the directory specified by the **-M** option.
- Regardless of how the **INI70** executable is started, the **-AN** option to the script overrides this behavior and directly specify the location of the **an_init.cfg** file. The Host (remote station) name must be listed in the [TCPIP] section.

```
[AISnet]
Protocol          = TCPIP
PrintErr          = 0
TraceLevel        = 0
LogFilePrefix     = ./an
#StderrLog        = stderr
No_Timeout        = 0
Host              = NISAW3
NrServer          = 10
MaxEnt            = 500
maxconnectretries = 5
t_connect_tries   = 2
t_connect_timeout = 250
t_snd_timeout     = 250
[TCPIP]
NISAW2            = "192.131.111.42 55555 /dev/tcp 1024"
NISAW3            = "192.131.111.48 55555 /dev/tcp 1024"
NISAW4            = "192.131.111.59 55555 /dev/tcp 1024"
[IAServer Connections]
1AW51A          = NISAW1 TCPIP 1 0 1 0 / 1
1AW51B          = NISAW2 TCPIP 1 0 1 0 / 1
1AW51C          = NISAW3 TCPIP 1 0 1 0 / 1
1AW51E          = NISAW4 TCPIP 1 0 1 0 / 1
```

There are times when NetFoxAPI/NetAIM*API tracing is required. To enable this, revise the indicated lines as follows:

```
PrintErr        = 2
TraceLevel      = 1
```

NOTE

The TCP/IP device name (**/dev/tcp**) has been changed from the one placed in the file by default (**\rtm\tcp**). The default device name is the one used for Microsoft Windows Operating Systems.

*Appendix B. NetFoxAPI/ NetAIM*API Licensing*

*This appendix explains the process of obtaining the required NetFoxAPI/*NetAIM*API *License.*

Overview

The NetFoxAPI/NetAIM*API software and license is included as part of the INI deliverable. The NetFoxAPI/NetAIM*API license codes can be updated with a new temporary code or permanent code without restarting the INI because the NetFoxAPI/NetAIM*API reads the license code only on startup.

The Host ID information needed to fill out the Request for Authorization Codes form on page 109 can be found by using the FoxAPI/AIM*API test program by typing this command from a DOS Command prompt **`d:\opt\foxind\ais\bin\foxtst.exe`**. Type a `-1` to exit the program.

— NOTE

Once the Request for Authorization Codes form (page 109) is completed it can be e-mailed to **aimauthorization@invensys.com**. Although the turnaround time for license requests is usually quite good, please allow up to 24 hours for a new license.

If e-mail is not available, fax the completed form to (**508**)-**549**-**4114** but be aware that the response may be delayed by an additional 24 hours.

The following shows an example of the information returned by Invensys when a NetFoxAPI/NetAIM*API license is issued:

```
Host ID:                        72715598
Server Code:                    BEUEPNOX
Machine Code for (2) Clients:   GTnemujcQujEndgwv

Host ID:                        808fd346
Server Code:                    CKVCDHFV
Machine Code for (2) Clients:   HRkjncddGvhN4thuQ

Host ID:                        80717dfc
Server Code:                    CKUEVBAD
Machine Code for (2) Clients:   Hxksmwhcuvmundhub

Host ID:                        8036db5b
Server Code:                    CKMNDXGC
Machine Code for (2) Clients:    HwkkdcfyTvhShnhuR
```

This excerpt shows the information supplied for each of four machines.

The first line for each machine is its Host ID. The second line is the Server Authorization code. This code is not needed for the NetFoxAPI/NetAIM*API software and can be ignored. The third line is the code that authorizes the client programs. It is this code that is entered in **an_init.tcp** in the line that begins SUNSPARC_AISnet.

Request for Authorization Codes

Information Network Interface

To: Attention: e-mail: Fax Number:	Invensys AIM* Licensing Team aimauthorization@invensys.com (508) 549-4114
Customer Contact Name: Telephone Number:	
Company Name: Sales Order Number:	
FAX the Authorization Codes to: Contact Name: FAX Number:	
e-mail the Authorization Codes to: Contact Name: e-mail Address:	

Complete this table by adding the Host ID(s) for the purchased product. Note that some line items will require multiple Host ID entries.

Part Number	Product Description	OS on Remote Machine (NetFoxAPI/ NetAIM*API Host)	Host ID of Remote Machine (NetFoxAPI/ NetAIM*API Host)	Number of NetFoxAPI/NetAIM*API Hosts
Q0301ZS	INI70 Asymmetric Software for Two Systems	Windows		1 License (1 host with 1 client.)
Q0301ZT	INI70 Asymmetric Software for Five Systems	Windows		5 Licenses (5 hosts each with 1 client.)
Q0301ZU	INI70 Symmetric Software for Two Systems	Windows		2 Licenses (2 hosts each with 1 client.)
Q0301ZV	INI70 Symmetric Software for Five Systems	Windows		25 Licenses (5 hosts each with 5 clients.)
Q0302AC	Upgrade to INI70 Asymmetric Software for Two Systems	Windows		1 License (1 host with1 client.)
Q0302AG	Upgrade to INI70 Asymmetric Software for Five Systems	Windows		5 Licenses (5 hosts each with 1 client.)
Q0302BN	Upgrade to INI70 Symmetric Software for Two Systems	Windows		2 Licenses (2 hosts each with 1 client.)
Q0303AQ	Upgrade to INI70 Symmetric Software for Five Systems	Windows		25 Licenses (5 hosts each with 5 clients.)
Q0303AJ	Isolation Station Software	Windows		1 License (1 host with 1 client.)
Q0303AK	Upgrade to Isolation Station Software	Windows		1 License (1 host with 1 client.)

Appendix C. Process Alarm Model

This appendix discusses Process Alarm handling.

A General Process Alarming Model

A general Process Alarming Model can be visualized in Figure C-1:

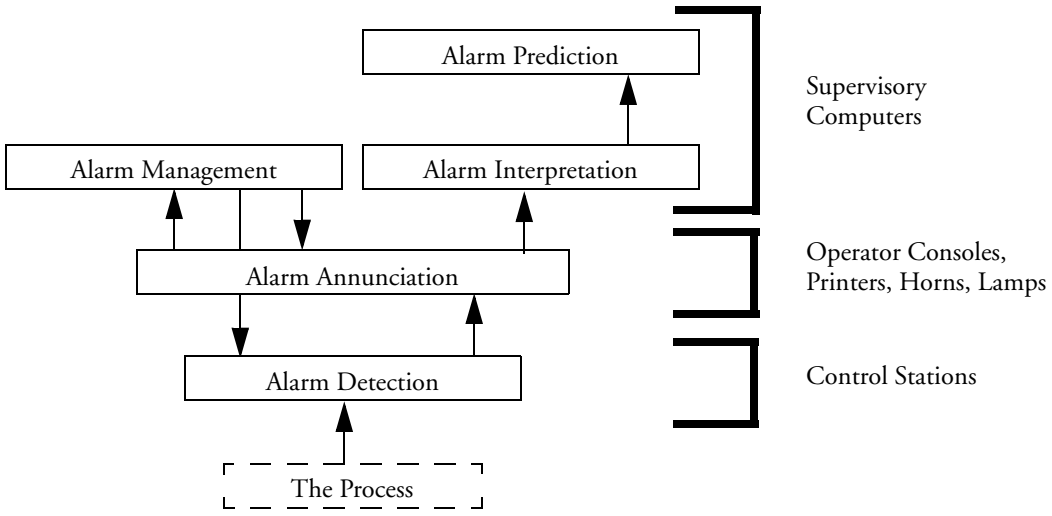


Figure C-1. General Process Alarming Model

The left-hand side of this picture shows the general levels of a Process Alarm System, and the right-hand side shows the implementation platforms within a generic control system.

For *Alarm Detection* to occur, the process must be analyzed to determine which values should be alarmed and the criticality or priority of each alarm must be decided. Instrumentation must be in place for those readily measurable alarm conditions, and a method of inferred calculation must be provided for those that are not readily measurable.

There are two main types of alarms: Analog and Digital. Analog Alarms include Bad I/O Alarms, Absolute Alarms, Deviation Alarms, and Rate of Change Alarms. Digital Alarms include Change of State Alarms, Pattern Alarms and Device Alarms.

Alarm Annunciation can take many different forms. Actions likely to result from the detection of one of the aforementioned alarm conditions might be the activation of a horn or flashing lamp, the printing of an error message at the printer, or the display of a flashing message at the operator console.

Alarm Management controls the detection, annunciation, presentation and historization of the process alarms.

Finally, the Supervisory Computers, with their expert process knowledge, are capable of understanding the causes of generated alarms, as well as predicting their occurrence. (*Alarm Interpretation and Prediction*)

Process Alarm Handling

Within the system, process control alarming is focused on the Control Blocks. The Control Block is the fundamental unit of Process Control within the system. Used together, a collection of Control Blocks implements all facets of a Process Control System. With regard to alarming, all Control Block algorithms follow a set of alarming standards. These standards include:

1. A fixed set of Alarm Types (mentioned above).
2. At most one instance of each Alarm Type per block.
3. A fixed set of Alarm Priorities.
4. Alarm Type Enabling standards and mechanisms.
5. Alarm Annunciation and Acknowledgment standards and mechanisms.
6. Alarm Inhibition standards and mechanisms.
7. Alarm Disabling standards and mechanisms.
8. Alarm Status presentation standards.

The Control Block Processor is a single program that implements all alarm functionality. Therefore maintaining these alarming standards is a relatively straightforward.

When an alarm condition is detected by the Control Block Processor, an alarm message is generated and sent to the devices specified in the device list associated with the alarming block.

At the same time, the block's **UNACK** (unacknowledged) parameter is set and the appropriate fields in the **ALMSTA** (alarm status) parameter are set.

The alarm remains unacknowledged until the operator or some program acknowledges the alarm by resetting the **UNACK** parameter in the alarming block.

When the **UNACK** parameter is reset, the Control Block Processor issues an Alarm Acknowledge Message. This message is sent to the same group of devices that received the initial alarm message.

Eventually, the block should return to normal. At that time, the Control Block Processor issues a **Return-To-Normal** message.

Figure C-2 illustrates the sequence of events:

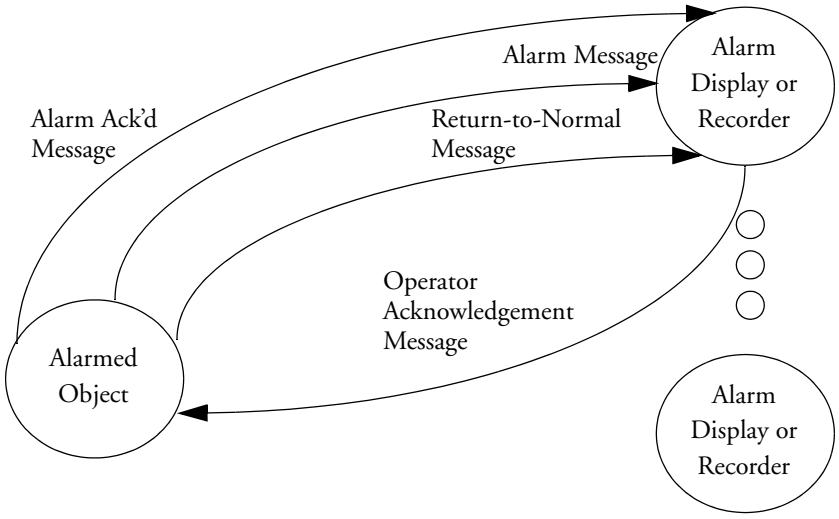


Figure C-2. Alarm Message Flow

Eventually, the block should return to normal. At that time, the Control Block Processor issues a Return to Normal message.

It is important to note that all of message traffic and hardware devices involved in this alarm-handling scheme exist on the same system.

Appendix D. Troubleshooting

This appendix provides troubleshooting techniques.

General Approach

The general approach to debugging any AOS based application is as follows:

1. Delete all of the log files in the **D:\opt\aos\logs** and any related log directories.
2. Repeat the AOS operation that failed.
3. Examine the log files.

Check the Network Connection and Configuration

The **D:\windows\system32\drivers\etc\hosts** file on the Remote Station must be modified to include both the IP address and the alias name associated with the second Ethernet port of the Local Station. Once this files has been modified, try to ping each machine from the other. Enter the following command from Local Station:

```
ping <remoteStationName>
```

The response should be:

```
<remoteStationName> is alive
```

Common Problems

Strange AOA Behavior

Application Object Attributes exist in a Shared Memory segment owned by the OM. This memory segment needs to be properly sized. If it is too small, one will get various strange behaviors, e.g., some A:O.As existing and other not and failure to open some OM list while other succeed.

Be sure to follow the instruction on page 26 and page 74 to ensure that the segment is properly sized.

Data Forwarding Specific

Check the NetFoxAPI/NetAIM*API Configuration Files

Ensure that the NetFoxAPI/NetAIM*API configuration files are properly setup. The following table gives some summary information.

Table D-1. Summary of Configuration Files

Station Where Used	Directory	File	Purpose	Station Where Closed	Created By	Refer to Section
Remote	**\opt\fox\ais\bin** **\opt\fox\aim\bin**	**foxapi.cfg** **aimapi.cfg**	Configures local FoxAPI or AIM*API limits.	Remote	User	4
Remote	**\opt\fox\ais\bin**	**an_init.tcp**	Configures access control to data on the Remote Station's Control Network.	Remote	an_setup	4
Local	One of: * Directory used to if the executable is started directly. * **D:\opt\aos\data** if the scripts are used to start the executable. * The directory specified by the **-M** option if the scripts are used to start the executable. * Specified by **-AN** option no matter how it is started.	**an_init.cfg**	Configures the source of remote data.	Remote	an_setup	4

Test NetFoxAPI Connection

Check that the NetFoxAPI connection is operational by using the **an_ping** program included as part of the NetFoxAPI client software delivered with the INI package.

The INI70 uses port number 55555 for FoxAPI. In the an_init.cfg on the local station, the respective line in an_init.cfg is a follows:

FoxAPI: (In this example 3AW70C is the remote station.)

```
[TCPIP]
:
3AW70C=”10.2.8.52 55555 /dev/tcp 1024”
:
[AIMServer Connections]
```

In the remote station’s /opt/aim/bin/an_init.tcp, ensure that all users are defined. For example:

```
[Security_Access]
administator=”System|Open Sets|W Files|W Objects|Read Only
jsmith=”System|Open Sets|W Files|W Objects|Read Only
fox=”System|Open Sets|W Files|W Objects|Read Only
```

Enter the following command from the Local Station:

```
cd d:/opt/netFoxAPIClient
an_ping <remoteMachineName>
```

The response should be as follows. *(In this example the Remote Station’s name is 5AW70A.)*

```
The machine type is "DOSPC"
The machine name is "1AW70C"
The user name is "Fox"
The Server "5AW70A" is alive.
This user "Fox" has the following access to AIS services on host
"5AW70A":
   Read files
   Write files
   Read objects
   Write objects
   Open data sets
   Close data sets
   All   AIS functions
   DOSDM    Not Authorized
   IASERVER Not Authorized
   FOXDM    Not Authorized
Fox API Version 4.3.1 Copyright (c) Foxboro 1992-1996
Toolkit is DOSPC
```

Test NetAIM*API Connection

Check that the NetAIM*API connection is operational by using the **an_ping** program included as part of the NetAIM*API client software delivered with the INI package.

The INI70 uses the port number 45678 for AIM*API (45678). In the an_init.cfg on the local station, the respective line in an_init.cfg is a follows:

AIM*API: (In this example 5AW70C is the remote station.)

```
[TCPIP]
:
5AW70C=”10.2.8.72 45678 /dev/tcp 1024”
```

```
:
[AIMServer Connections]
```

In the remote station's /opt/aim/bin/an_init.tcp, ensure that all users are defined. For example:

```
[Security_Access]
administator="System|Open Sets|W Files|W Objects|Read Only
jsmith="System|Open Sets|W Files|W Objects|Read Only
fox="System|Open Sets|W Files|W Objects|Read Only
```

Enter the following command from the Local Station:

```
cd d:/opt/netFoxAPIClient
an_ping <remoteMachineName>
```

The response should be as follows. *(In this example the Remote Station's name is 5AW70C.)*

```
The machine type is "DOSPC"
The machine name is "6AW70B"
The user name is "ffung"
The Server "5AW70C" is alive.
This user "ffung" has the following access to AIS services on host
"5AW70C":
   Read files
   Write files
   Read objects
   Write objects
   Open data sets
   Close data sets
   All   AIS functions
   DOSDM    Authorized
   IASERVER Authorized
   FOXDM    Authorized
AIM API Version 5.50 Copyright (c) Foxboro 1992-1996
Toolkit is DOSPC
```

General Diagnostic Techniques

When a repeatable problem occurs, do the following:

- Stop the INI if possible.
- Remove all log files from **D:\opt\aos\logs**.
- Remove all logs from the directory specified by using the **-L** option of **INIMAP** or **INIMMAP**.
- Repeat the operation that caused the problem.
- Examine the log files. Look for lines that contain the string "ERR", "error", or "failed". If **grep**(**1**) is used specify the **-i** option to do a case insensitive search.

Information to Gather Before Reporting a Problem

When the INI is having problems, take the following steps:

- If the application currently exists, remove it using **INIDELETE**.
- Remove all files from **D:\opt\aos\logs**.
- Build the application using **INIBUILDDB**.
- Create the objects using **INICREATE**.
- Map the application using **INIMAP** or applications using **INIMMAP** (Note: the **-L** option allows the specification of a particular log directory).

- If the problem occurs during mapping, watch for the problem and then turn on the debugging trace using **INIDEBUG**.

At this point you should have seen your problem and the information required to debug it should be in the log files.

Then send the following files:

- The **VERSION** file from **D:\opt\aos**.
- The map file or files.
- The various log files.
- The commands as you typed them.

In the e-mail message describe as best you can the behavior that has been seen. Be sure to explain when in the process described above the problem occurred.

If you are mapping multiple applications in a single INI instance, all of the map files must be sent. It is also important to send the build and create logs for each application. This means that after reach run of **INIBUILDDB** and **INICREATE**, the log files will have to be collected since they would normally overwrite each other.

This information should allow the support people to duplicate the problem.

A Read-Only Application - Example

Goal

The goal of test is to establish that the INI is capable of reading a value from a Remote Station and updating it on a regular basis.

Procedure

Create an environment that has the following menu items linked as specified:

Menu Item Label	Location	Description
AOSTst	**D:\opt\aos\examples\Tests**	The Tests directory contains the displays required to run the tests.
AOSTst	**D:\opt\aos\Monitor\disp**	The disp directory of Monitor contains displays that can be used to monitor the performance of the application when it is mapping.

The menu hierarchy is as follows:

AOSTst
- **AOS_Tests** (see Figure D-1)
  - **MultiMap** (see Figure D-2)
    - **MULTI_APP** (see Figure D-4)
  - **Standard** (see Figure D-2)
    - All of the standard tests (see Figure D-3)

INI_Tests (see Figure D-1)

All of the standard tests (see Figure D-5)

The following figures show these menus:

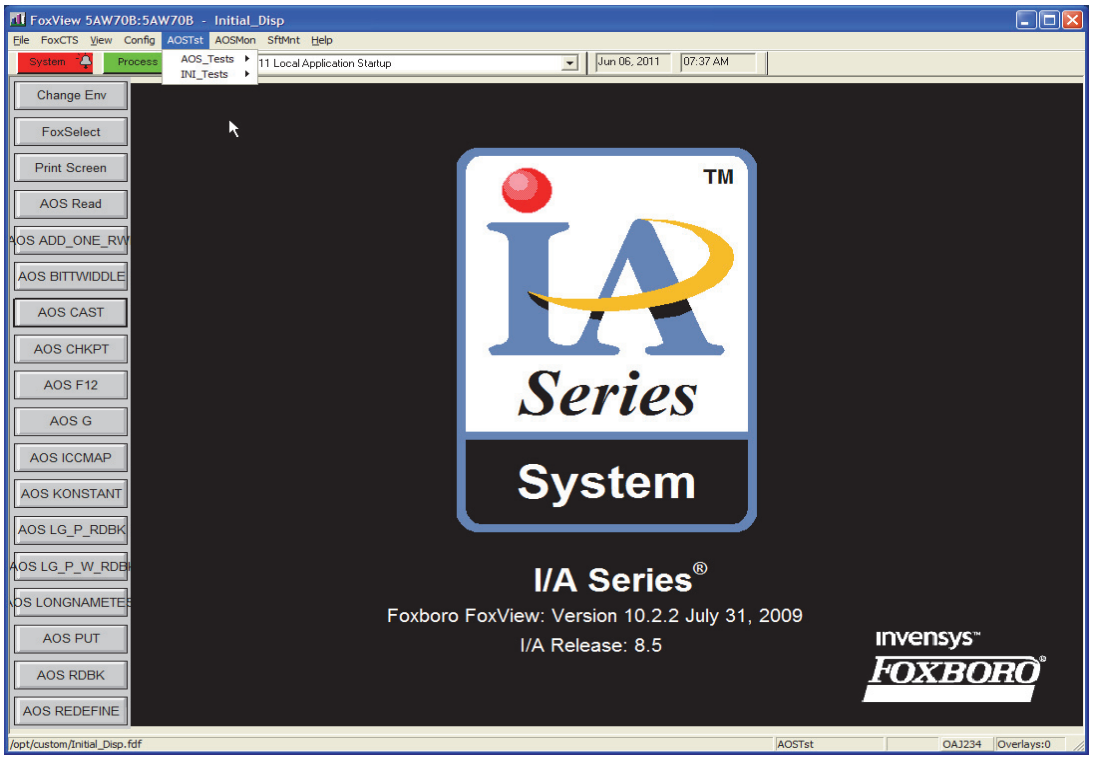


Figure D-1. Contents of the AOSTst Menu Item (Click on AOSTst)

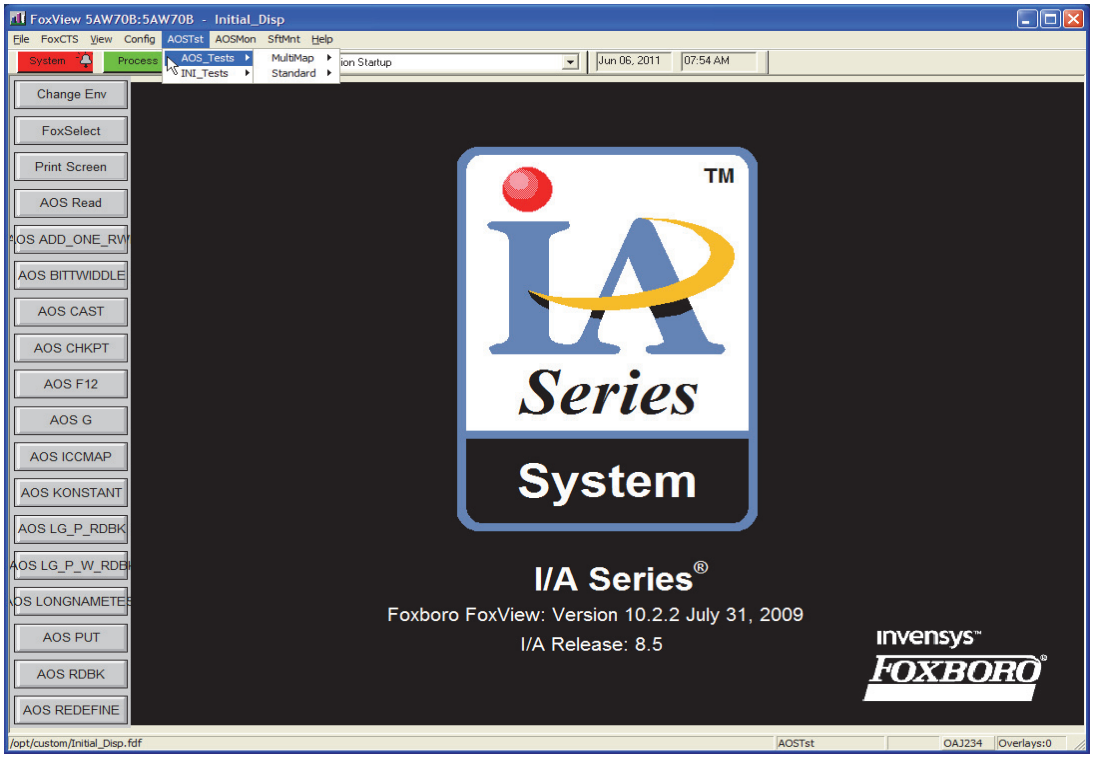


Figure D-2. Contents of the AOSTst

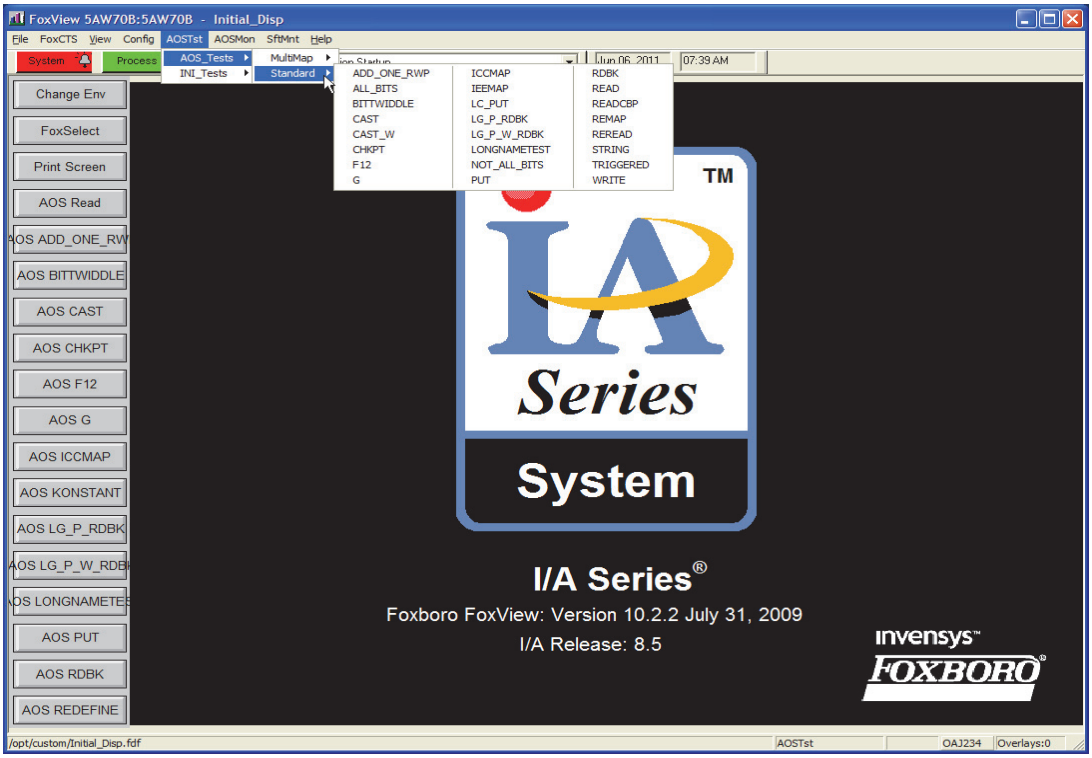


Figure D-3. Standard AOS Tests

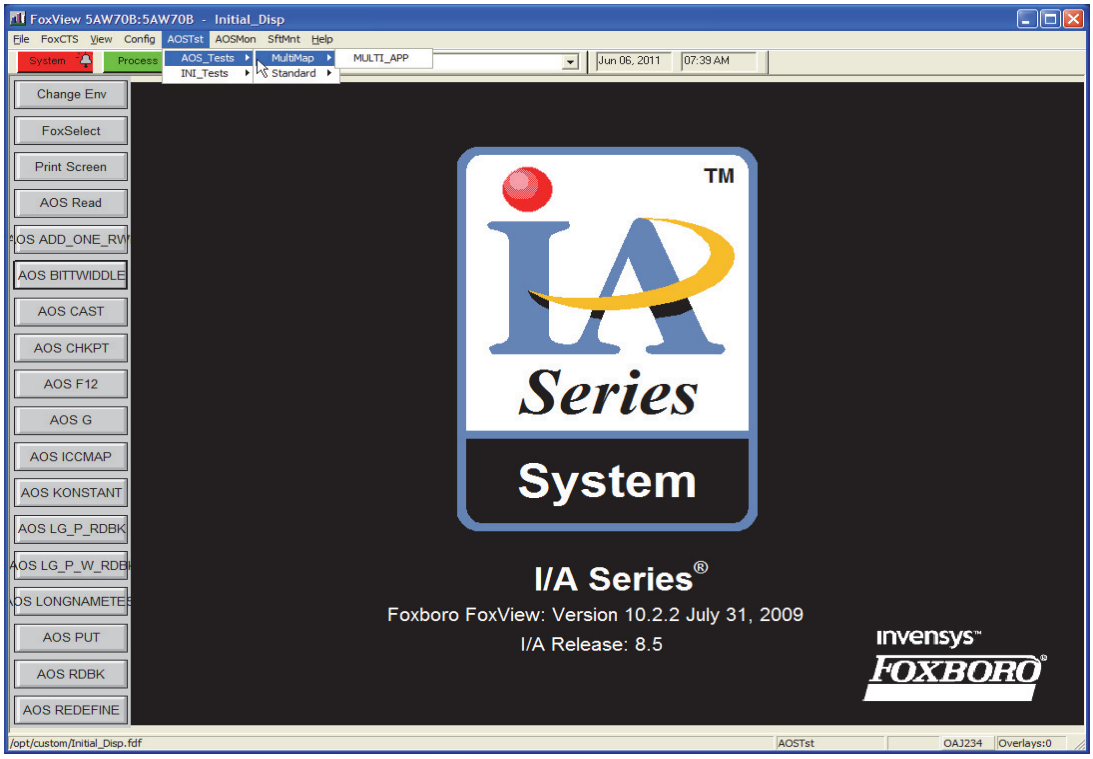


Figure D-4. Multiple Application Single Mapping Process Test

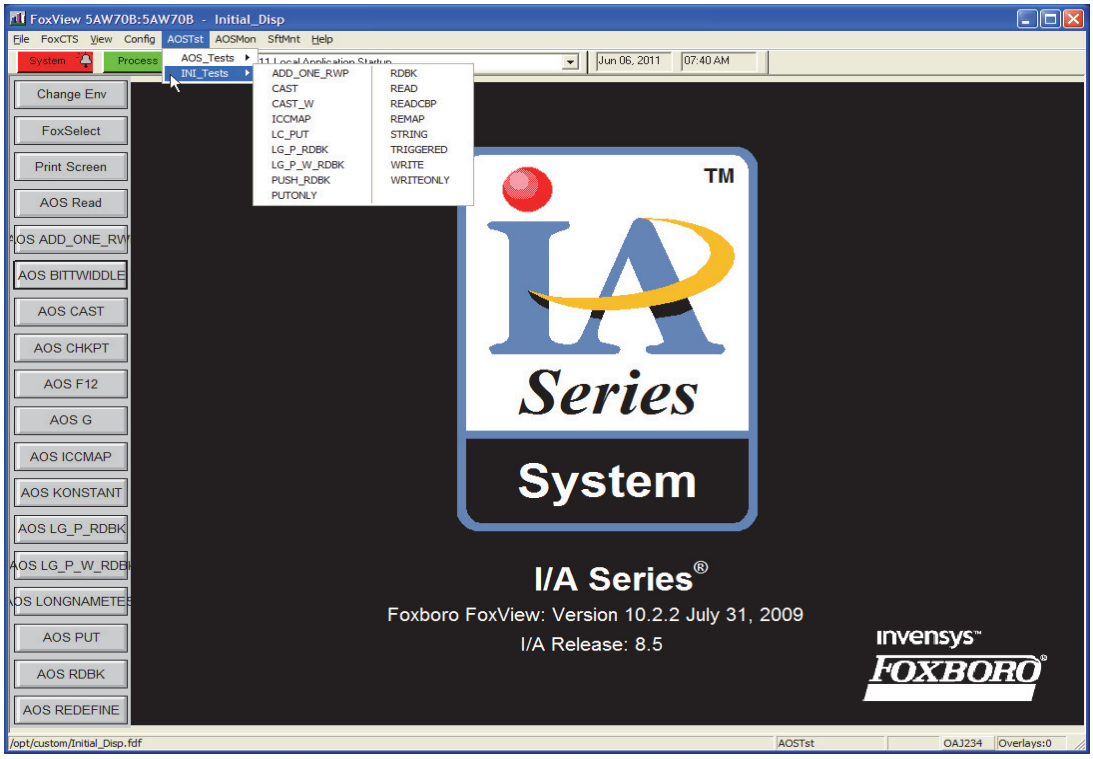


Figure D-5. Standard INI Tests

Use the **AOSTst** menu in the INI testing environment to raise the INI **READ** test display. The steps are:

- Select `AOSTst` from the main menu bar.
- Select `INI_TEST` from the AOSTest menu and then select `READ` from the menu items.

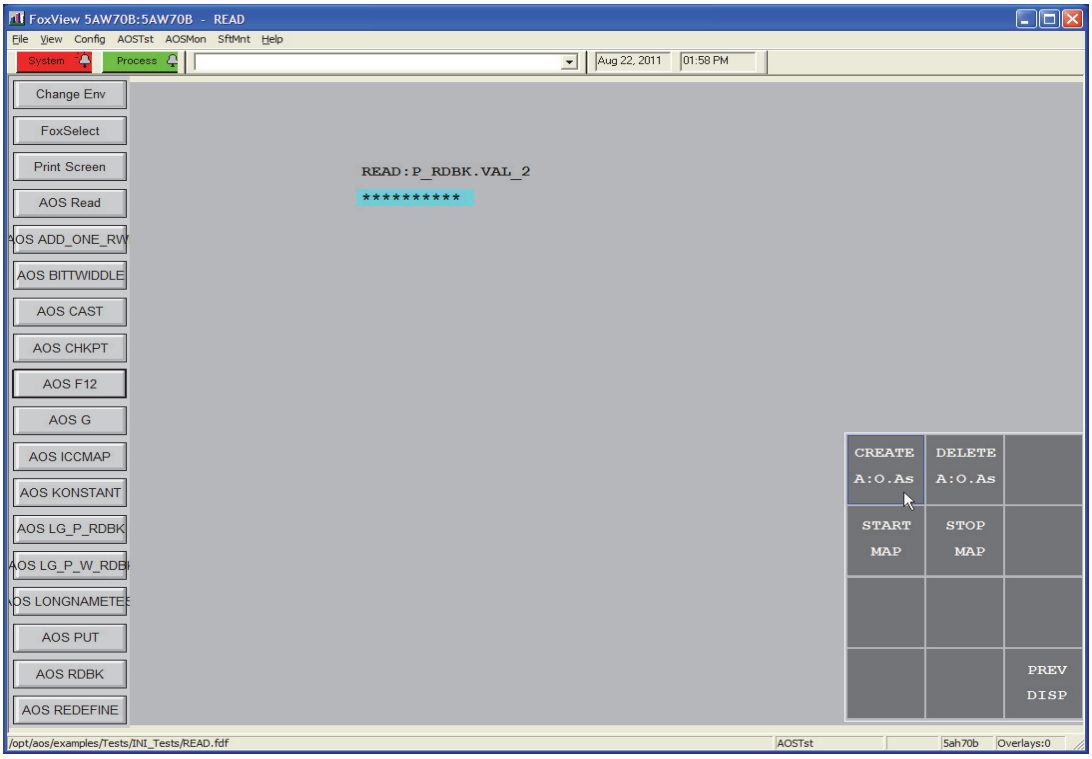


Figure D-6. READ Test Display

Once the display is up, take the following steps:

- Replace the **an_init.cfg** file in **D:\opt\aos\examples\ini\READ\data** with the one used to test the NetFoxAPI/NetAIM*API connection - see page 117.
- Create the required Application Objects in both the local and Remote Stations using the button labeled **CREATE AOAs**. The objects are created when the message "**AOAs Created.**" appears on the message line.
- Start the INI mapping process by pressing the button labeled **START MAP**. When the message, "**Mapping Started.**" appears on the message line, the testing can begin.
- From the Remote Station, change the value of the Application Object in the Remote Station by typing:

  D:/opt/fox/bin/tools/omset -i <newValue> READ:P_RDBK.VAL_2

- Observe the display on the Local Station, the value should change in about 2 seconds.
- Change the value on the Remote Station and observe the result on the Local Station several times.

Criteria For Success

The test was successful if the value on the Local Station tracks the value on the Remote Station.

Message Forwarding

Check Proper Operation

In this example the second Ethernet port of the Remote Station is named **cr1aw1** and the second Ethernet port of the Local Station is named **cr2wp1**.

The following assumes that the Message Forwarding Client and Remote Station software has been installed and configured per the instructions in this manual.

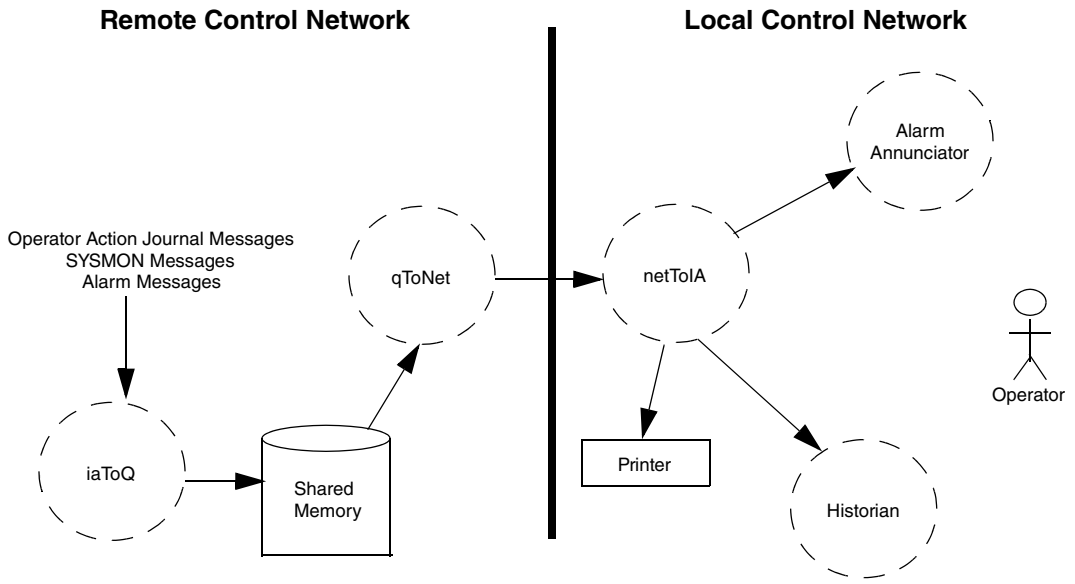


Figure D-7. Message Forwarding - Local and Remote Stations

On the Local Station

- Edit the file **D:\opt\aos\MsgFwd\data\FileOfNames** to include the following line. For this test we creates a group named **MSGFWD**.

  ```
  MSGFWD RH HORN:1.IN HORN:2.IN NONE NONE NONE HORN:S.IN
  ```

- Because we have edited the **FileOfNames** file we have to stop Message Forwarding.

  ```
  cd D:/opt/aos/MsgFwd/scripts
  stop_MsgFwd
  ```

- Now restart Message Forwarding so that the modified file `FileOfNames` gets re-read.

  D:/opt/aos/MsgFwd/exe/netToIA

- Create a Message Forwarding test compound named **Horn**.
- Create six test **COUT** blocks in the compound **Horn**. These test blocks should be named **1**, **2**, **3**, **4**, **5**, and **S** and the **IOMOPT** parameter should be set to **0**.
- Place be blocks in Automatic and turn on the compound.
- Enter the Detail Display for Block 1. Notice that this block displays **FALSE**.

On the Remote Station

- The Message Forwarding Remote Station software has been installed and configured per the instructions in this manual.
- Edit the file **D:/opt/aos/MsgFwd/data/FileOfNames** to match the **FileOfNames** file on the Local Station. For the test we creates a group named **MSGFWD**.

  ```
  MSGFWD RH HORN:1.IN HORN:2.IN NONE NONE NONE HORN:S.IN
  ```

- Because we have edited the **FileOfNames** file we have to stop Message Forwarding.

  ```
  cd D:/opt/aos/MsgFwd/scripts
  stop_MsgFwd
  ```

- Now restart Message Forwarding so that the modified file **FileOfNames** gets re-read.

  ```
  D:/opt/aos/MsgFwd/exe/iaToQ IATOQ <LocalStation>
  ```

- Use the script **/opt/aos/MsgFwd/scripts/testAlm** to test Message Forwarding.

  ```
  cd /opt/aos/MsgFwd/scripts
  testAlm MSGFWD A
  ```

On the Local Station

- When the message is received on the client system, the Detail Display for Block 1 will display **TRUE**.
- The following steps can be used to reset the Block to attempt the test again.
  - Select the **Config**.
  - Select the **In**.
  - Select **Toggle**.

If no message was received, check your configuration and make sure that you have configured Message Forwarding properly.

Message Relay

Check Proper Operation

To test the implementation of Message Relay, the following steps should be taken:

- Follow the steps on page 79 to ensure that the network is operating properly.
- Open four windows on the Local Station. Position the windows so that there are two on the top and two on the bottom of the screen.
- In the two right most windows, map to the D: drive of the Remote Station.
- In the lower right most window, do the following to start a simple server:

```
cd /opt/aos/MsgRelay/data
../exe/rcv RCV -D5
```

- In the upper right most window, do the following to start the Message Relay process:

```
cd /opt/aos/MsgRelay/data
../exe/msgRelay MSGRELAY nisaw3 -p FileOfTargets -D5
```

- In the upper left most window, do the following to start the Message Relay process:

```
cd /opt/aos/MsgRelay/data
../exe/msgRelay MSGRELAY nisaw2 -D5
```

- Wait for the programs running in the upper windows to stabilize before taking the next step. The delay should be at least 30 seconds after the last message is printed.
- In the lower left most window, do the following to start a simple client:

```
cd /opt/aos/MsgRelay/data
../exe/snd RCV SND -D5
```

As the test runs, you should see messages move from the `snd` process on the Local Station to the `rcv` process on the Remote Station and vice versa.

Any problems should be clearly visible in the traces found in the four windows.

snd - Message Relay Test Client

Sends messages to the Remote Station

SYNOPSIS

```
snd <targetName> <iaName> [alias [...]] [-local] [-bcast]
```

where:

<targetName> - Remote Station's logical/device name. This name must be unique on the Remote Station's network. It should also be one of the names in the configuration file for the local copy of **msgRelay**.

<iaName> - The logical name for this instance of the snd program. This name should be in the configuration file for the remote copy of **msgRelay**.

<alias> - The program will send a message for each specified alias. These names should be in the configuration file for the remote copy of **msgRelay**.

-local - Tells the program to register its name within the station, but not globally with the OM.

-bcast - Specifies that the program should use a broadcast to send its message.

DESCRIPTION

The program **snd** exists on the Local Station. It exists to demonstrate the operation of **msgRelay**. It operates in conjunction with the program rcv which runs on the Remote Station.

This pair of programs can be used to demonstrate the proper operation of the Message Relay component of the INI package.

EXAMPLE

There are three modes of operation supported by the Message Relay component:

- Case 1 - Act as a proxy for programs with well-known names, i.e., those registered with the OM.
- Case 2 - Relay a broadcast message from one system to the next.
- Case 3 - Act as a proxy for programs with locally known names assuming that the client program meets certain criteria.

Case 1 is run with **snd** as follows:

```
D:/opt/aos/MsgRelay/exe/snd RCV SND -D5
```

In this test, the Local Station's **FileOfProxies** must contain one entry:

```
RCV
```

In this test, the Remote Station's **FileOfProxies** must contain one entry:

```
SND
```

Both programs (**snd** and **rcv**) will register with the OM.

Case 2 is run with **snd** as follows:

```
D:/opt/aos/MsgRelay/exe/snd RCV SND -D5
```

In this test, the Local Station's **FileOfProxies** must contain one entry:

```
RCV BCAST
```

In this test, the Remote Station's **FileOfProxies** must contain one entry:

```
snd
```

The **rcv** program must be setup with the **-local** option.

Case 3 is a little more involved:

```
D:/opt/aos/MsgRelay/exe/snd RCV SND -local -D5
```

In this test, the Local Station's `FileOfProxies` must contain one entry:

```
RCV
```

or

```
RCV BCAST
```

depending on if the **rcv** process has the **-local** option or not.

In this test, the Remote Station's **FileOfProxies** must contain one entry:

```
SND
```

rcv - Message Relay Test Server

Receives messages to the Local Station and replies to them.

SYNOPSIS

```
rcv <iaName> [<alias> [...]] [-local] [-bcast]
```

where:

<iaName> - The logical name for this instance of **rcv**. This name should be in the configuration file for the Local Station's instance of **msgRelay**.

<alias> - The program may be addressed by each specified alias. These names should be in the configuration file for the Local Station's instance of **msgRelay** if they are to be used.

-local - Tells the program to register its name within the workstation, but not globally with the OM.

-bcast - Specifies that the program should use a broadcast to send its message.

DESCRIPTION

The program **rcv** exists on the Remote Station. It exists to demonstrate the operation of **msgRelay**. It operates in conjunction with the program snd which runs on the Local Station.

This pair of programs can be used to demonstrate the proper operation of the Message Relay component of the INI package.

EXAMPLE

There are three modes of operation supported by the Message Relay component:

- Case 1 - Act as a proxy for programs with well-known names, i.e., those registered with the OM.
- Case 2 - Relay a broadcast message from one system to the next.
- Case 3 - Act as a proxy for programs with locally known names assuming that the client program meets certain criteria.

Case 1 is run with **rcv** as follows:

```
D:/opt/aos/MsgRelay/exe/rcv RCV -D5
```

In this test, the **FileOfProxies** for the Local Station's instance of **msgRelay** must contain one entry:

```
RCV
```

and both programs (**snd** and **rcv**) will register with the OM.

Case 2 is run with **rcv** as follows:

```
D:/opt/aos/MsgRelay/exe/rcv RCV -D5
```

In this test, the **FileOfProxies** for the Local Station's instance of **msgRelay** must contain one entry:

```
RCV BCAST
```

In this test, the **FileOfProxies** for the Remote Station's instance of **msgRelay** must contain one entry:

```
SND BCAST
```

or

```
SND
```

The choice is made based on the **-local** setting for **snd**.

Case 3 is setup as follows:

```
D:/opt/aos/MsgRelay/exe/RCV -D5
```

In this test, the Local Station's **FileOfProxies** must contain one entry:

```
RCV
```

In this test, the Remote Station's **FileOfProxies** must contain one entry:

```
SND
```

Index

D

E

F

G

I

L

M

N

O

P

Q

R

S

T

U

Invensys Systems, Inc.
10900 Equity Drive
Houston, TX 77041
United States of America
http://www.invensys.com

Global Customer Support
Inside U.S.: 1-866-746-6477
Outside U.S.: 1-508-549-2424
Website: https://support.ips.invensys.com