Advanced Simulation Technology inc.
500 A Huntmar Park Drive
Herndon, Virginia 20170 U.S.A.
Tel. (703)471-2104 • Fax. (703)471-2108
www.asti-usa.com

ASTi ACE Studio Technical User Guide

Document: DOC-01-TELAS-UG-4

Advanced Simulation Technology inc. 500A Huntmar Park Drive, Herndon, Virginia, 20170 USA
Revision Q (March 2013)

Product Name: ACE Studio

ASTi ACE Studio Technical User Guide

ASTi

500-A Huntmar Park Drive

Herndon, VA 20170

Table of Contents

1.0. Introduction

ACE represents the latest evolution of ASTi's sound and communications model development, debug, and runtime environment. Designed to accommodate the distributed nature of today's simulation systems and architectures, ACE offers a new and powerful method of model development and maintenance for both single and networked ASTi devices.

ACE Studio is a suite of software tools incorporating sound and communications model development, debug and management, communications monitoring and fault analysis, and equipment status and configuration. ACE Studio provides remote access to all networked simulation models and equipment from a central location and runs as an application on approved customer workstations.

ACE Studio features include:

- Remote access to any ASTi model or equipment over LAN/WAN
- Real-time model development, management, and debug
- Centralized exercise and communications planning and monitoring
- Equipment status and configuration
- Legacy equipment upgrade support
- Security enhancements per DISA STIGs

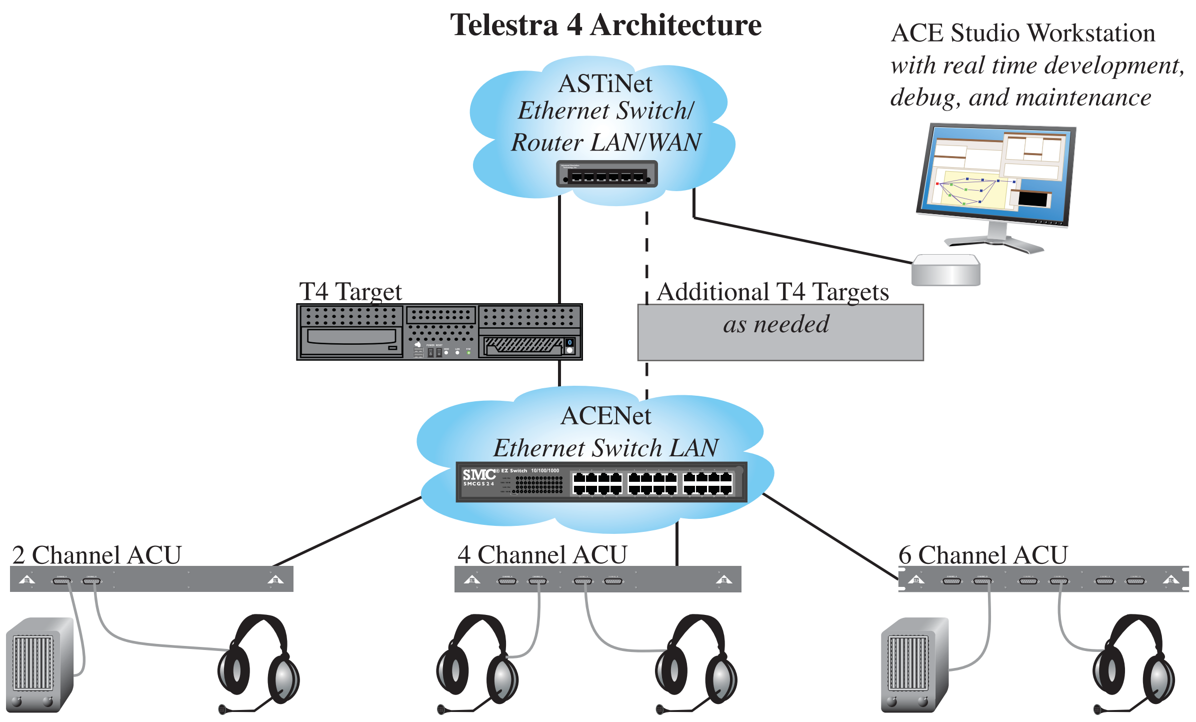


Figure 1: Telestra 4 Architecture

1.1. Telestra 4 Concept of Distributed Model Development

In the years since the DACS system was first developed, training systems have evolved. In general, the trend has moved toward more complex, more capable applications, often involving higher-level requirements such as multiple (including coupled) trainer modes, large operator counts, HLA support, Text-to-Speech and Speech Recognition capabilities, etc.

In many ways, T4-ACE was designed to solve a more complex problem, and is optimized to manage these additional complexities. The addition of what may seem like added levels of complexity enables the T4 system to:

- Manage multiple Targets from a central location with the ability to reconfigure them on the fly to meet the demands of today's trainers.
- Solve larger, complex, and evolving problem spaces such as managing an entire facility verses a simple one-piece trainer solution.
- Solve complex applications using a one-box solution with added features such as HLA and other server applications requiring additional configuration.
- In the Project Manager, quickly piece together reusable elements to build models using Helpers. The Helpers remove the need to dig down into the model level for simple cookie-cutter models.
- Solve a wide array of training and simulation applications.

While at a first glance these layers may seem complex, the ACE Studio additional layers are intended to maximize the reuse of model elements, enhance management and provide portability.

In ACE Studio, a Project consists of several layers of audio system hardware, software models, and network configuration parameters. ASTi created these layers of information to extract all networking configuration and hardware specifics from the model, which allows the model to be changed on the fly without having to reconfigure the parameters.

In ACE Studio, there are several layers to become familiar with. The first layer in a Project is the Layout which contains the Project's configuration. Each Layout assigns the resources to the Load. These resources include domains, comm plans, and sound repositories, etc. The Load consists of sets of models created in ACE Model Builder. The model layers are similar to past ASTi simulation models with parameters and primitives to drive the components.

See section 3.0. for more information on Projects.

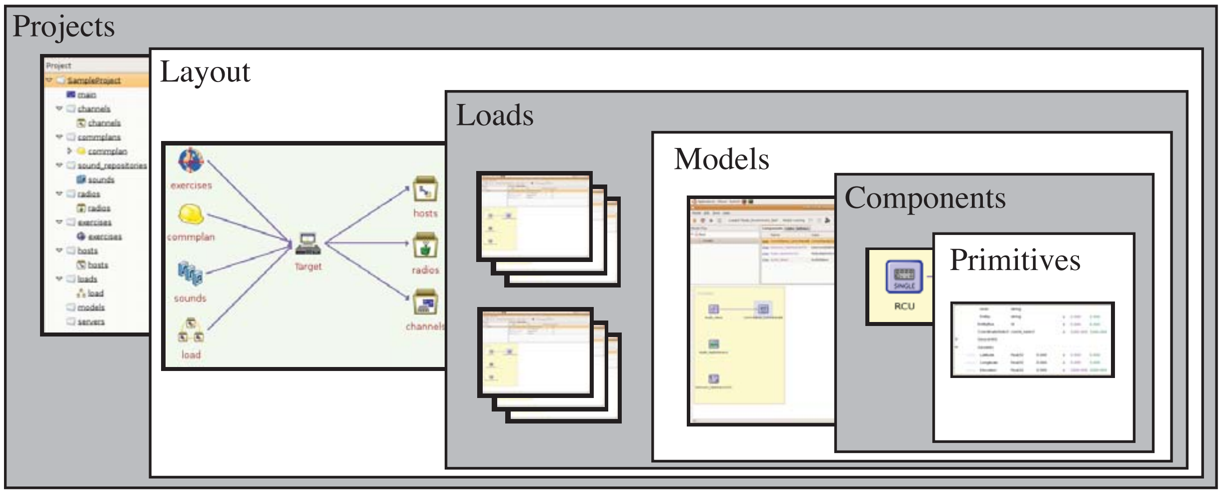
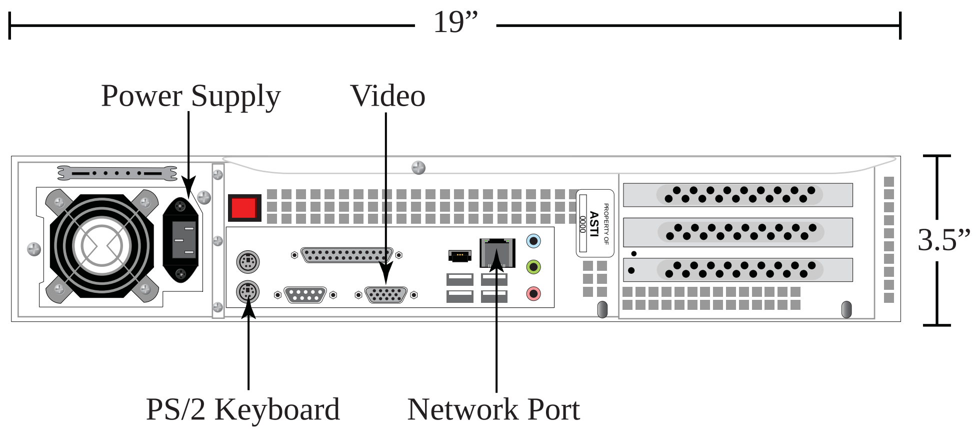


Figure 2: Project Layers

2.0. System Installation

ACE Studio software runs on an ASTi-provided Development Workstation or on a customer-provided system running a virtual machine.

The Development Workstation is a 19” 2U rackmount platform configured with a single ethernet port, removable hard drive and DVD drive. Follow the steps in the Telestra 4 Quick Start Guide (DOC-01-TEL4-QSG-1) for initial system setup and IP address assignment.



2.2. Software Installation

The Development Workstation comes pre-configured. To install the ACE Studio software from scratch see the ACE Studio Cold Start Procedure (DOC-02-TEL4-ASCS-1).

To install ACE Studio on a virtual machine see the ACE Studio VM Quick Start Guide (DOC-01-ASVM-QSG-4).

2.2.1. Default Network Settings

The system does not have a default IP address. Follow the steps in the Telestra 4 Quick Start Guide (DOC-01-TEL4-QSG-1) for initial system setup and IP address assignment.

2.3. System Accounts and Services

The only user account that exists after system installation or cold start is "root" (without the quotes). The default password for this user is also "abcd1234." All user account logins and passwords are case-sensitive.

```
Username: root         Password: abcd1234
```

For normal ACE Studio operation use:

```
Username: aceuser      Password: aceuser
```

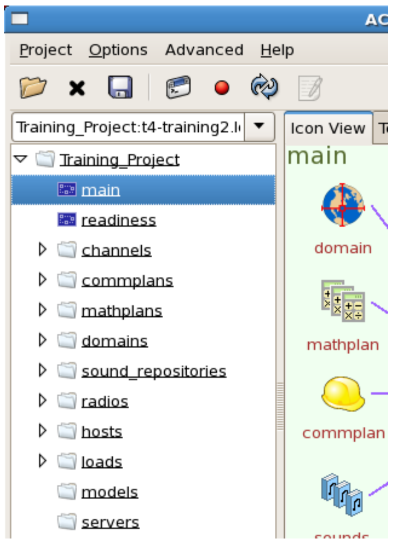
Warning: ASTi strongly recommends changing the 'aceuser' password after coldstart.

3.0. ACE Studio -The Big Picture

3.1. Projects

Many of today's simulation and training applications have transitioned beyond simple, stand-alone training devices to multi-platform, complex, networked simulation applications. Projects provide the ability to develop, configure, and manage sound and communications models, simulation applications, and other related elements across a set of platforms and applications. Projects can manage greater simulation complexities and allow successful interoperation across platforms.

A Project is a sound and communications simulation scenario consisting of a combination of hardware (e.g. modeling platforms, audio and I/O distribution, simulation servers), simulation software (e.g. sound and communications models, SATCOM, Terrain, Datalink), and configuration elements (communications plans, entity assignments, exercise parameters).



A Project in its simplest form can represent the sound and communications hardware, software, and models for a simple stand-alone desktop simulator. On the opposite end of the spectrum, a project can encompass many training devices and applications participating in a WAN-based simulation architecture or exercise.

3.1.1. Project Elements

Projects elements include:

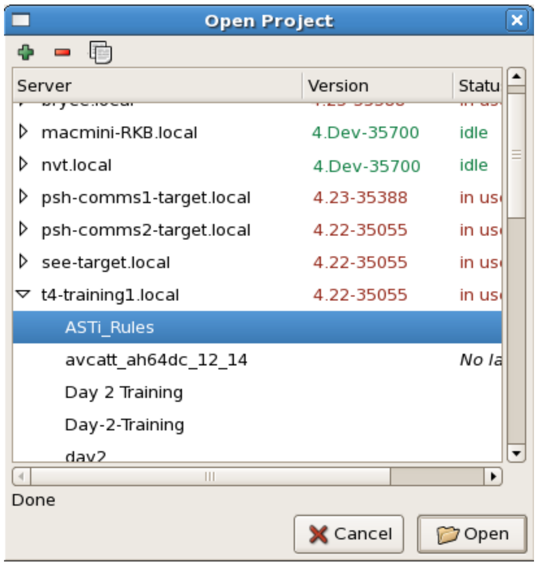
- **Targets** – embedded modeling platforms that run sound and communications models and other ASTi simulation applications.
- **Audio and I/O Distribution Devices** – ASTi's ACENet Communication Units (ACUs), ACU2s, ACE-RIUs and ACENet compatible audio amplifiers.
    - ACUs, ACU2s and ACE-RIUs provide remote digital audio and I/O distribution between Targets and audio peripherals (e.g. military and commercial headsets, powered speakers, tape units, DVRs, and real world communications equipment). Distribution is via ASTi's ACENet protocol over dedicated Ethernet-based networks.
    - ACENet compatible audio amplifiers are used for sound reinforcement in environmental cue applications. These amplifiers support direct connection to the ACENet distribution architecture eliminating the need for an individual audio I/O device and audio amplifier in environmental cue applications.
- **Servers** – ACE server software runs server-based simulation applications and services such as SATCOM, Terrain, high-frequency (HF) propagation environments, HLA, Datalink, and NTP.
- **Communications and Sound Models** – Communications and sound model elements developed by the user can be distributed, linked, and managed as part of the Project. Models are developed using ACE Studio's model generation tools.
- **Sound Repositories** – Recorded sound libraries used by sound and communications models are developed and managed as part of the Project.
- **Host Interface Configuration** – Setup and configuration of host interfaces to sound and communications models are provided as part of Project development. This approach helps users develop modeling elements that are reusable across platforms and are agnostic to any particular host simulation software's structure.
- **Communications Plans** – Using ACE Studio's communications planning tool, radio, intercom, and other communications related assets can be configured and managed across a set of models and applications to help ensure interoperability and allow users to change, store and reuse communications parameters for different exercises.
- **Domain Editor** – Exercise related parameters such as entity assignments, DIS, and HLA parameters are managed as part of the Project.
- **Target Loads** – Configuration of models, simulation applications, host interfaces, and other elements for each Target are managed as part of the Project.

3.2. Project Manager Tool

Projects are developed and managed using the Project Manager tool. When launched, the Project Manager searches the network for available Targets. Each Target found is queried and a list is generated of existing Projects stored on each Target. The user can then pick to work with any Project from the available list. Alternatively, the user can elect to build a new Project and select to develop on any available Target. All Project development and modifications occur on the selected Target and are executed from that platform.

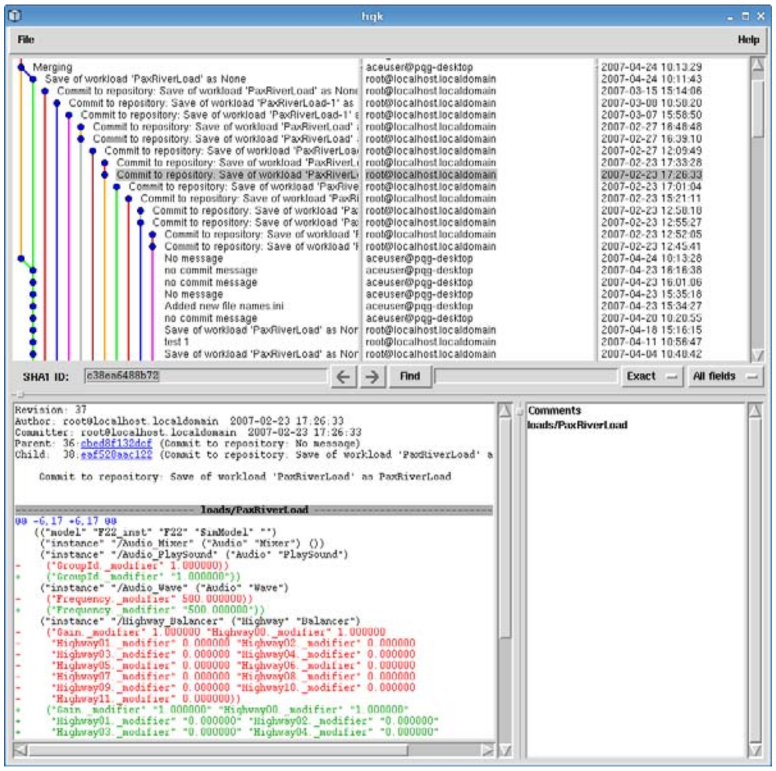
Within Project Manager the user can perform typical file operations on a Project such as Open, Close, New, and Duplicate. ***Remember that operations performed in a Project are done on the selected Target.***

Project development and changes are also tracked through a built-in control management system. This allows the user to manage Projects in a similar fashion as they would software source code. Features such as change tracking, change descriptions, release management and the ability to return work with earlier release instances provides powerful configuration management capabilities.



Server	Version	Statu
macmini-RKB.local	4.Dev-35700	idle
nvt.local	4.Dev-35700	idle
psh-comms1-target.local	4.23-35388	in us
psh-comms2-target.local	4.22-35055	in us
see-target.local	4.22-35055	in us
t4-training1.local	4.22-35055	in us
ASTi_Rules		
avcatt_ah64dc_12_14		No la
Day 2 Training		
Day-2-Training		
day2		

Below is an example of a View Log menu item selection in Project Manager. Note that the user can view the entire change history of their project.



```
Merging                                                 aceuser@pqg-desktop          2007-04-24 10:13:29
Save of workload 'PaxRiverLoad' as None                 root@localhost.localdomain   2007-04-24 10:11:43
Commit to repository: Save of workload 'PaxRiverLoad' as Non  root@localhost.localdomain   2007-03-15 15:14:06
Commit to repository: Save of workload 'PaxRiverLoad-1' as    root@localhost.localdomain   2007-03-08 10:50:20
Commit to repository: Save of workload 'PaxRiverLoad-1' a     root@localhost.localdomain   2007-03-07 15:58:50
Commit to repository: Save of workload 'PaxRiverLoad' a       root@localhost.localdomain   2007-02-27 16:48:48
Commit to repository: Save of workload 'PaxRiverLoad' .       root@localhost.localdomain   2007-02-27 16:39:10
Commit to repository: Save of workload 'PaxRiverLoa           root@localhost.localdomain   2007-02-27 12:09:49
Commit to repository: Save of workload 'PaxRiverL             root@localhost.localdomain   2007-02-23 17:33:28
Commit to repository: Save of workload 'PaxRiverL             root@localhost.localdomain   2007-02-23 17:26:33
Commit to repository: Save of workload 'PaxRive               root@localhost.localdomain   2007-02-23 17:01:04
Commit to repository: Save of workload 'PaxRi                 root@localhost.localdomain   2007-02-23 15:21:11
Commit to repository: Save of workload 'Pa                    root@localhost.localdomain   2007-02-23 12:58:18
Commit to repository: Save of workload 'Pa                    root@localhost.localdomain   2007-02-23 12:55:27
Commit to repository: Save of workload 'F                     root@localhost.localdomain   2007-02-23 12:52:05
Commit to repository: Save of workload 'F                     root@localhost.localdomain   2007-02-23 12:45:41
No message                                              aceuser@pqg-desktop          2007-04-24 10:13:28
no commit message                                       aceuser@pqg-desktop          2007-04-23 16:16:38
no commit message                                       aceuser@pqg-desktop          2007-04-23 16:01:06
No message                                              aceuser@pqg-desktop          2007-04-23 15:35:18
Added new file names.ini                                aceuser@pqg-desktop          2007-04-23 15:34:27
no commit message                                       aceuser@pqg-desktop          2007-04-20 10:20:55
Save of workload 'PaxRiverLoad' as Nor                  root@localhost.localdomain   2007-04-18 15:16:15
test 1                                                  root@localhost.localdomain   2007-04-11 10:56:47
Save of workload 'PaxRiverLoad' as Nor                  root@localhost.localdomain   2007-04-04 10:40:42
```

SHA1 ID: c38ea6488b72

```
Revision: 37
Author: root@localhost.localdomain  2007-02-23 17:26:33
Committer: root@localhost.localdomain  2007-02-23 17:26:33
Parent: 36:cbed8f132dcf (Commit to repository: No message)
Child:  38:eaf520aac122 (Commit to repository: Save of workload 'PaxRiverLoad' a

    Commit to repository: Save of workload 'PaxRiverLoad' as PaxRiverLoad

------------------------------ loads/PaxRiverLoad ------------------------------
@@ -6,17 +6,17 @@
   (("model" "F22_inst" "F22" "SimModel" "")
    ("instance" "/Audio_Mixer" ("Audio" "Mixer") ())
    ("instance" "/Audio_PlaySound" ("Audio" "PlaySound")
-     ("GroupId._modifier" 1.000000))
+     ("GroupId._modifier" "1.000000"))
    ("instance" "/Audio_Wave" ("Audio" "Wave")
-     ("Frequency._modifier" 500.000000))
+     ("Frequency._modifier" "500.000000"))
    ("instance" "/Highway_Balancer" ("Highway" "Balancer")
-     ("Gain._modifier" 1.000000 "Highway00._modifier" 1.000000
-      "Highway01._modifier" 0.000000 "Highway02._modifier" 0.000000
-      "Highway03._modifier" 0.000000 "Highway04._modifier" 0.000000
-      "Highway05._modifier" 0.000000 "Highway06._modifier" 0.000000
-      "Highway07._modifier" 0.000000 "Highway08._modifier" 0.000000
-      "Highway09._modifier" 0.000000 "Highway10._modifier" 0.000000
-      "Highway11._modifier" 0.000000))
+     ("Gain._modifier" "1.000000" "Highway00._modifier" "1.000000"
+      "Highway01._modifier" "0.000000" "Highway02._modifier" "0.000000"
+      "Highway03._modifier" "0.000000" "Highway04._modifier" "0.000000"
```

```
Comments
loads/PaxRiverLoad
```

Some Project management features are also available through ASTi's Remote Management System (RMS) web services. Pointing a network browser to a Target from any convenient computer connected to the same network accesses the web-based RMS application on that Target.

Within RMS, the user selects the Project Management tab. From the pages under this tab the user can view local and global Projects.

Local Projects (Projects present on the local Target that RMS is pointing to):

- Display list of Projects on the current Target
- Backup Projects from the Target
- Delete Projects from the Target
- View Change Logs of each Project

Global Projects (Projects on other machines visible to the Target over the network):

- Display list of Projects on other Targets
- View change Logs of each Project on other Targets

Additionally

- Copy new projects on a Target
- Clone new projects from any Target

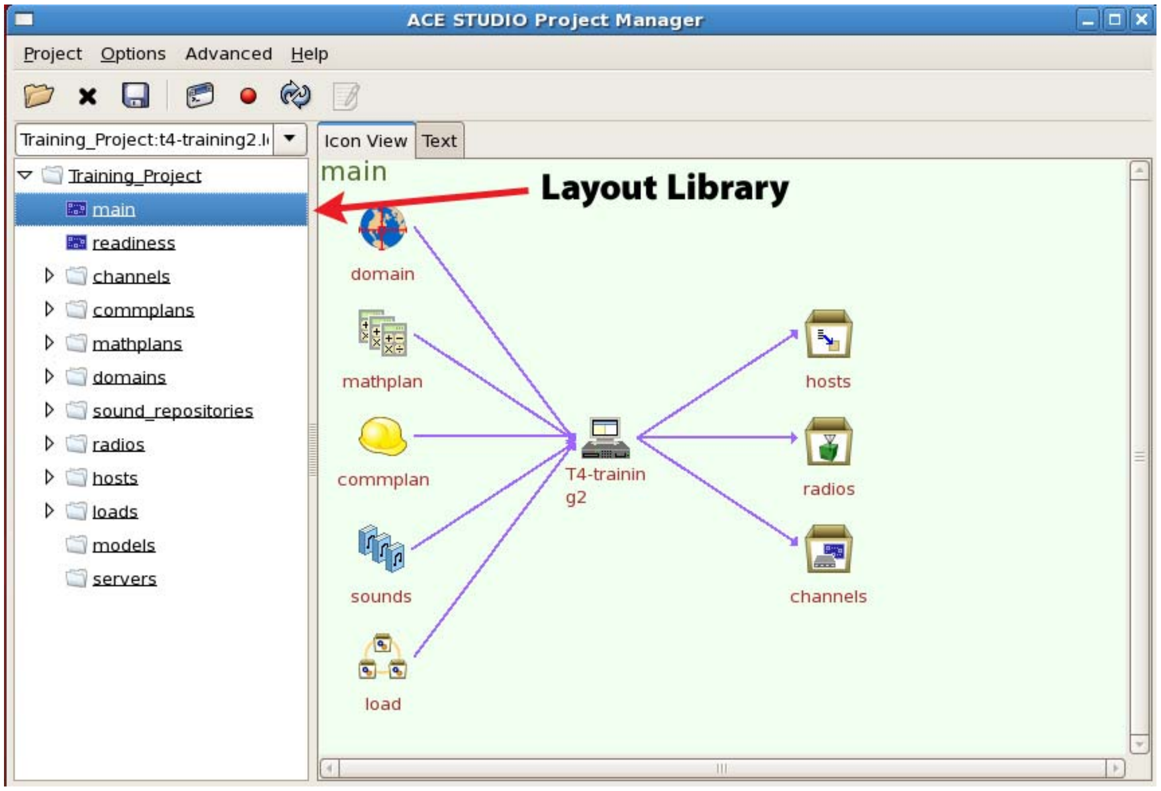
Note: For more information on Project capabilities in RMS, see the RMS 4 User Guide (DOC-01-TEL4-RMS4-UG-1).

3.3. Layout

A Layout contains the Project's configuration. The Layout consists of a collection of user- and tool-generated elements such as hardware, models, interfaces, communications assets, exercise and communications planning parameters. For veteran ASTi users, the Layout is comparable to the configuration (CFG) file on the DACS.

Using the Layout graphical and text-editing tools, the developer selects, links and configures these elements from the current Project libraries to create a Layout. The developer also has the option of adding and generating new elements for the Project, which may or may not be used as part of the Layout.

Links between icons show dependencies and associations of the individual Project elements. For example, a link from a Load element to a Target element indicates the Load will be installed and run on that particular Target. A communication plan may be linked to a Target indicating that it will use the communications plan after executing the Layout. A layout is needed for each Target on the network.



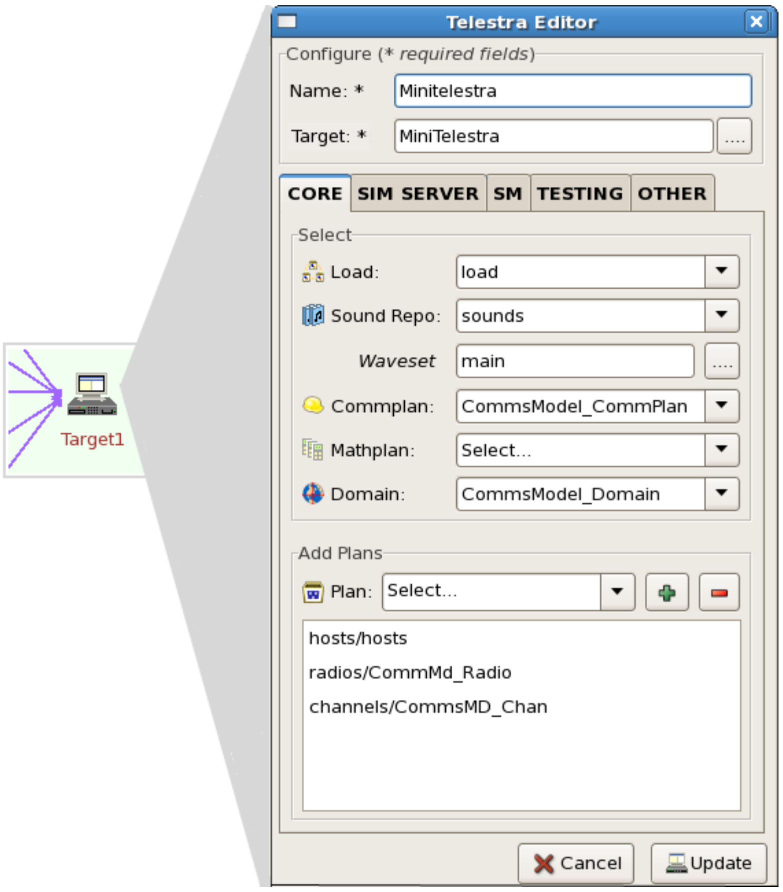
3.3.1. Target Setup in the Layout

The user can add or edit a Target by double-clicking in the default Layout on the Target icon. In the 'Telestra Editor' the user must select the configuration items that are set in Project Management. The name field and Target field are filled automatically. The Load name must be specified.

The 'Core' tab provides the general setup for all Targets. The waveset is a folder that contains the wave files, this must be set before creating sound files in the Sound Repository.

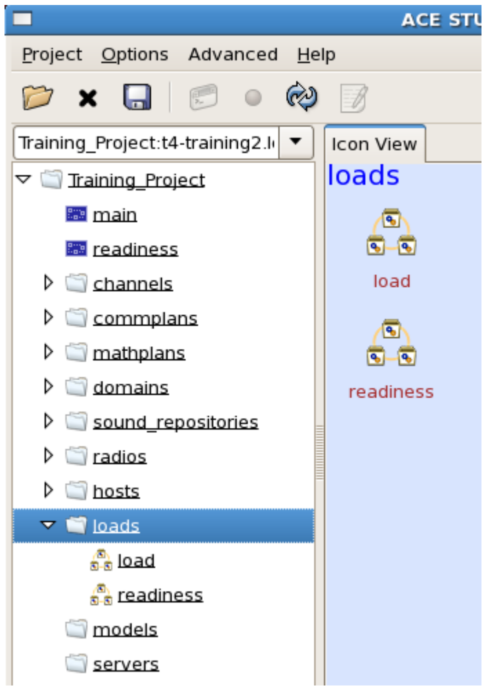
The set up of the comm plan, math plan, and Domain are optional.

Set the DIS Gateway under the Sim Server tab. This is required for DIS communications.



Name: * Minitelestra
Target: * MiniTelestra
Load: load
Sound Repo: sounds
Waveset main
Commplan: CommsModel_CommPlan
Mathplan: Select...
Domain: CommsModel_Domain
Plan: Select...
hosts/hosts
radios/CommMd_Radio
channels/CommsMD_Chan

3.4. Loads

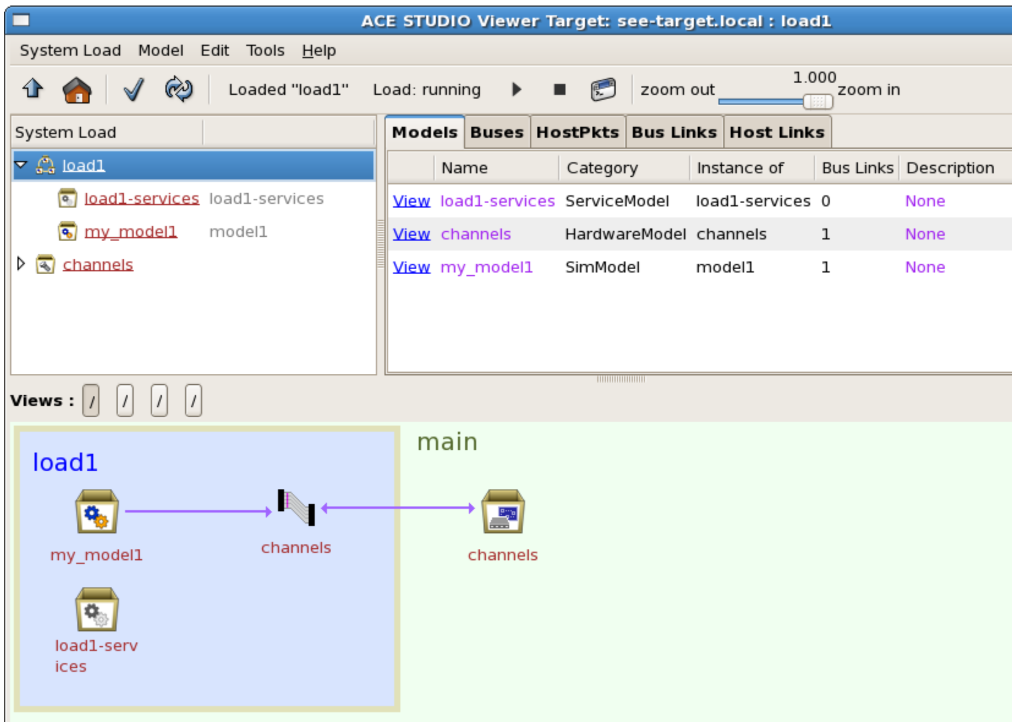


A Load is a collection of models that have been built and linked together to form an overall communications and sound model to run on a selected Target. Models can be generated via ACE Studio's model development tools, or generated by the various helpers or a combination of both.

An empty Load is generated in the Project Manager by selecting the Loads folder, right-clicking on the canvas and selecting 'Add' from the pop-up menu. The models contained in the Load are then created using the ACE Studio model generation tool. A Load can also be created from scratch within the ACE Studio model generation tool.

To apply a Load to a Target, double-click on the desired Target in the Layout view. The Target configuration window will open with a Load drop down list which allows the user to select a Load to run on a Target.

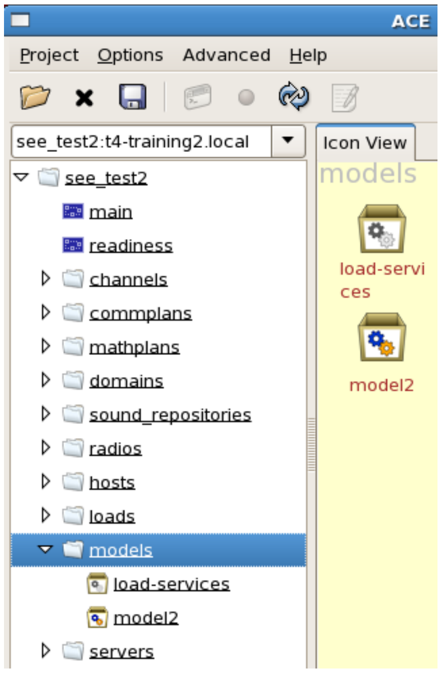
Note: The user should understand the difference between a Project and a Load. A Load is a model set for a specific Target whereas a Project is a complete configuration of Loads, Comm Plans, Servers, etc. across one or more Targets, Servers and simulation applications.


ACE STUDIO Viewer Target: see-target.local : load1
System Load Model Edit Tools Help
Loaded "load1" Load: running
zoom out 1.000 zoom in
System Load
load1
load1-services load1-services
my_model1 model1
channels
Models Buses HostPkts Bus Links Host Links
Views :
load1
main
my_model1
channels
channels
load1-serv ices

	Name	Category	Instance of	Bus Links	Description
View	load1-services	ServiceModel	load1-services	0	None
View	channels	HardwareModel	channels	1	None
View	my_model1	SimModel	model1	1	None

3.5. Models

Models are the individual modeling elements generated by either Helpers and/or ACE Studio's modeling environment. A model can be small and simple, such as a set of components that model an engine sound, a ship board stereo operator, or an F-16 Caution Warning system. A model can also be large and complex such as the entire communications system for an F-18 platform.



Models are self-contained and can be linked together; therefore, the user can create a Library of reusable model components to build larger, more complex models.

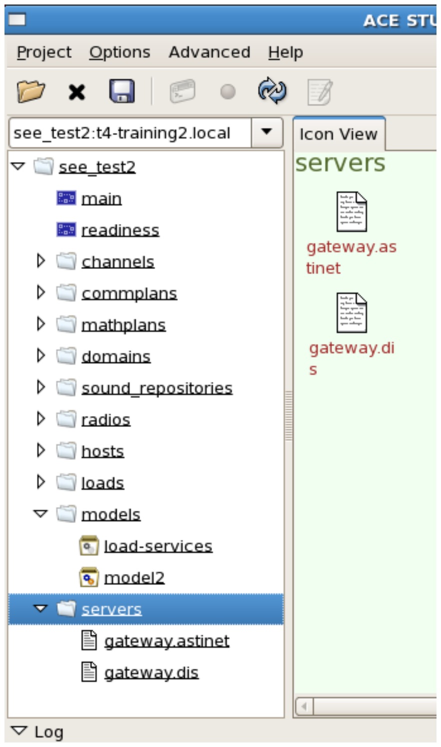
All models that are built by the Project Helpers or by ACE Studio's model development tools can be added to a Project. At a minimum, all models used by the Targets contained in the Project will be visible in the Model's folder. Additionally, models can be added to the Project to create a library of reusable components whether or not they are used by the current Project's Layout.

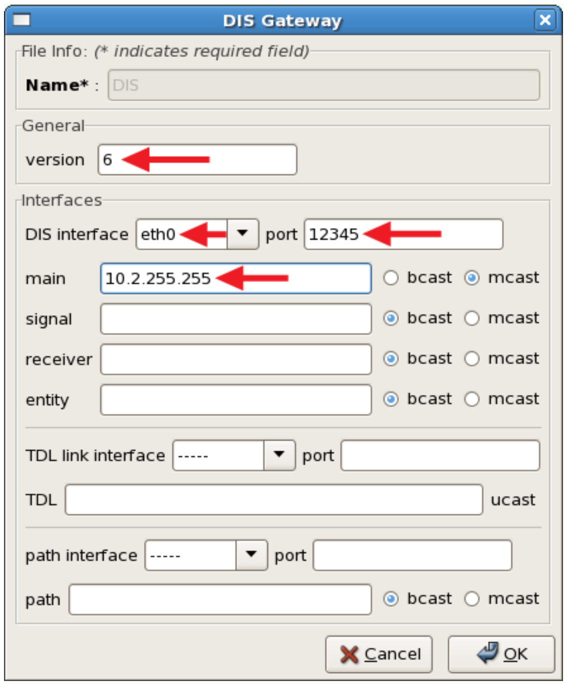
3.6. Servers

Servers provide a configuration of advanced software options setup for a specific Target known as the gateway. The gateway external settings are most commonly used for setting up DIS or HLA parameters.

In the DIS Gateway, the user adds the information necessary for the system to run on the DIS network including:

- Version 4, 5, 6, or 7
- Interface eth0, eth1 through ethN
- UDP Port Number
- Main Broadcast or Multicast Address





File Info: *(* indicates required field)*

Name* : DIS

General

version 6

Interfaces

DIS interface eth0 port 12345

main 10.2.255.255 bcast [✓ mcast]

signal [✓ bcast] mcast

receiver [✓ bcast] mcast

entity [✓ bcast] mcast

TDL link interface ----- port

TDL ucast

path interface ----- port

path [✓ bcast] mcast

3.6.1. DIS

Distributed Interactive Simulation (DIS) is a simulation protocol standard developed jointly by industry and the military to enable interoperation of simulation and training devices over local and wide area networks.

One of the more difficult and often underestimated aspects of simulation over local and wide area networks is achieving a realistic radio communication environment. With the DIS option active, the local radio and intercom modeling performed by the Target software is extended over the local and wide area network. Communication simulation between multiple DIS-compatible network devices is invisible to the user with full radio modeling across systems. All recently released versions of the DIS standard are supported and available to the user for selection.

During DIS operation, the Target transmits and receives DIS standard PDUs. Since the Target is involved strictly with communications simulation it is only concerned with Transmitter, Signal and Receiver PDUs.

The exception to this is Entity State PDUs, which are received to accommodate entity attach features whereby a radio modeled on the Target is attached to an entity on the network.

3.6.2. Host Control of DIS Gateway Settings

In software version 4.39-37894 and later, a user can change the following settings via ssh host control:

- Version 4, 5, 6, or 7
- Interface eth0, eth1 through ethN
- UDP Port Number
- Main Broadcast or Multicast Address

The script assumes that a project and DIS gateway file already exist. Running the script makes the change to the file and saves it to the project. The result is the same as if the changes were made in Studio directly. Follow the steps below to change DIS Gateway Settings via ssh host control:

1. Close ACE Studio.
2. Ssh to the T4 Target platform in question. For details on ssh refer to the following:

   www.asti-usa.com/support/appnotes/99.html
3. Execute the following command:

   ```
   ace set-dis-gw <projname> <disfilename.dis> version=<n>
   port=<nnnn> interface=eth<n> main_[b|m]cast=<nnn.nnn.n.n>
   ```

   where:

   `<projname>` = T4 Project name as seen in ACE Studio and RMS

   `<disfilename.dis>` = DIS Server Gateway file name as seen in ACE Studio

   `version=<n>` = DIS Version

   `port=<nnnn>` = UDP Port

   `interface=eth<n>` = Interface

   `main_[b|m]cast=<nnn.nnn.n.n>` = Main Broadcast or Multicast Address

   Note: the settings (version, port, etc) do not start with a dash. The command will print out warnings and errors if there are any.
4. *Important:* A project reinstall is required for the changes to take effect.

The example that follows is for a project named “NASMP”, a DIS Gateway file named “FST”, DIS Version 5, Interface eth0, and a multicast address of 231.1.1.1.

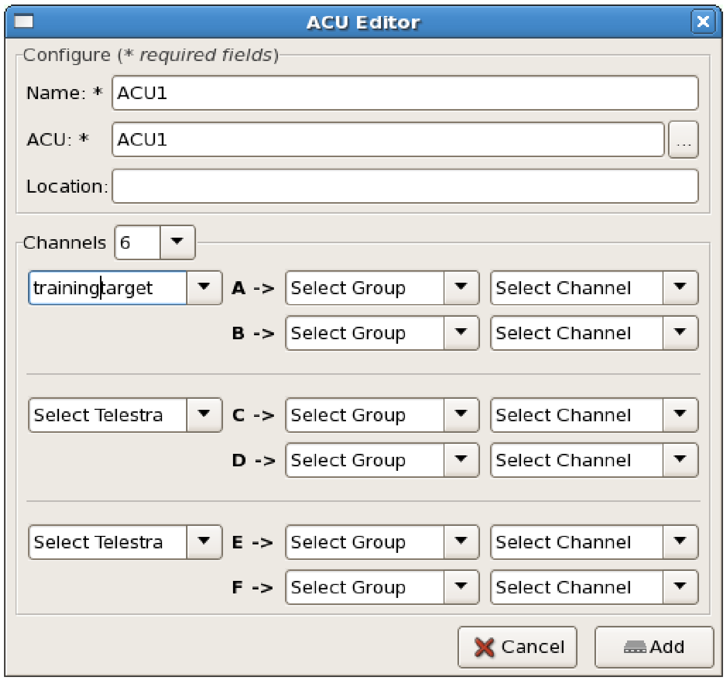
Example

```
[admin@telestra-DC-C0-5E ~]$ ace set-dis-gw NASMP FST.dis
version=5 port=12349 interface=eth0 main_mcast=231.1.1.1

requesting all changes adding changesets adding manifests
adding file changes added 1 changesets with 23 changes to 23
files updating working directory 23 files updated, 0 files
```

```
merged, 0 files removed, 0 files unresolved No username found,
using 'admin@…' instead pulling from  https://localhost/
projects/NASMP searching for changes no changes found pushing
to  https://localhost/projects/NASMP searching for changes
adding changesets adding manifests adding file changes added 1
changesets with 1 changes to 1 files

Updated version,port,interface,main_mcast settings in the dis
file FST.dis. Please reinstall the project

[admin@telestra-DC-C0-5E ~]$ ace-user install-layout NASMP main

Layout installed on localhost
```

3.7. Audio Distribution Devices

ASTi's ACENet protocol provides audio distribution over dedicated Ethernet-based networks. ACUs, ACU2s and ACE-RIUs provide remote digital audio and I/O distribution between Targets and audio peripherals (e.g. military and commercial headsets, powered speakers, tape units, DVRs, and real world communications equipment).



Name: * ACU1

ACU: * ACU1

Location:

Channels 6

trainingtarget A -> Select Group, Select Channel

B -> Select Group, Select Channel

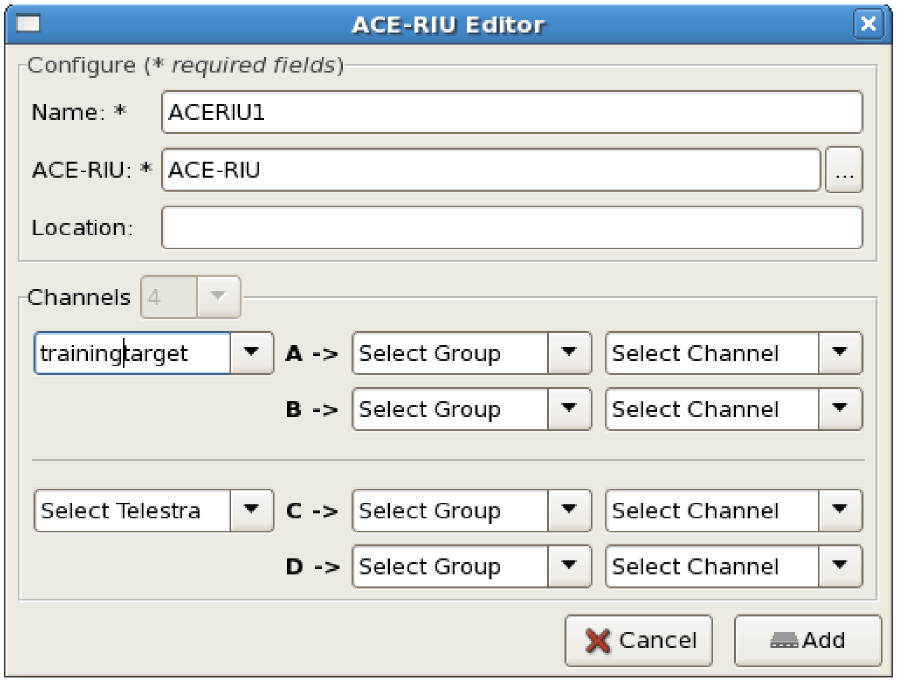
Select Telestra C -> Select Group, Select Channel

D -> Select Group, Select Channel

Select Telestra E -> Select Group, Select Channel

F -> Select Group, Select Channel

The ACENet audio distribution devices including the ACU, ACU2, ACE-RIU, and the ACENet compatible Crown® Amp are configured via the Remote Management System and ACE Studio. The model must have an I/O component to route audio out of the audio distribution device.


ACE-RIU Editor

Configure (* *required fields*)

Name: * ACERIU1

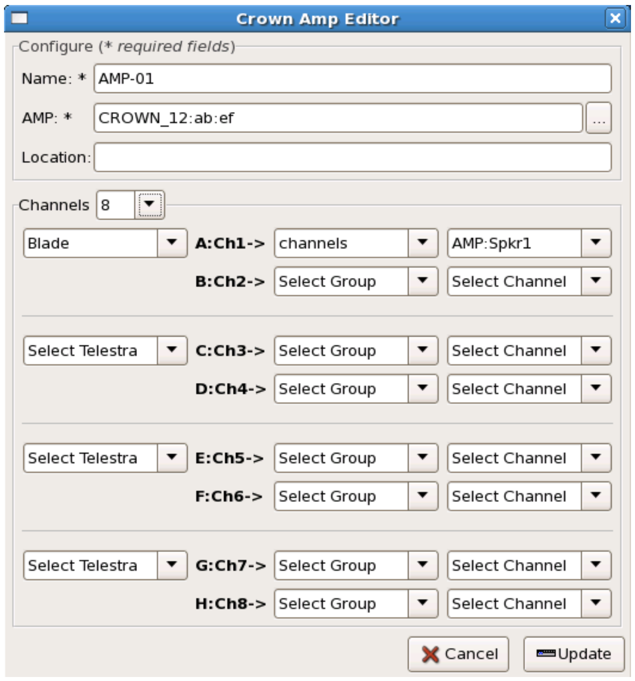
ACE-RIU: * ACE-RIU

Location:

Channels 4

trainingtarget	A ->	Select Group	Select Channel
	B ->	Select Group	Select Channel
Select Telestra	C ->	Select Group	Select Channel
	D ->	Select Group	Select Channel

Users can add any audio distribution devices connected to the ACENet network to their Project by right-clicking in the Layout canvas. To configure the audio distribution devices, right-click the icon to open the 'Editor' window. Each device is given a unique name that must be entered. The user must specify the designated Target, group, and channel.


Crown Amp Editor

Configure (* *required fields*)

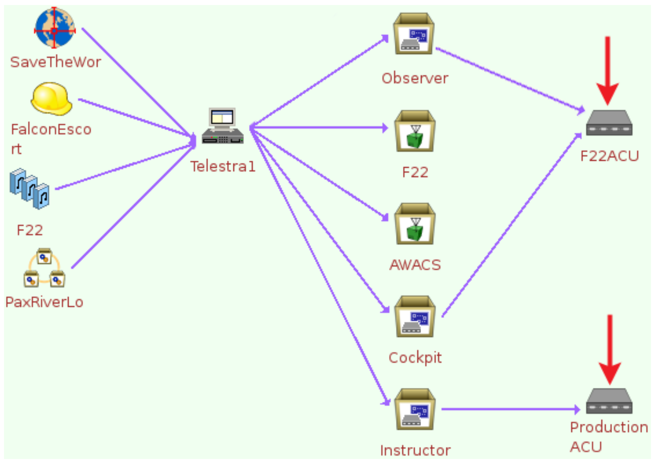
Name: * AMP-01

AMP: * CROWN_12:ab:ef

Location:

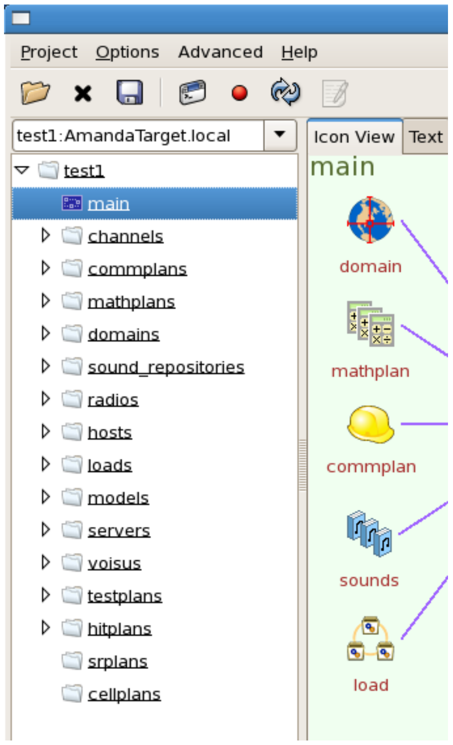
Channels 8

Blade	A:Ch1->	channels	AMP:Spkr1
	B:Ch2->	Select Group	Select Channel
Select Telestra	C:Ch3->	Select Group	Select Channel
	D:Ch4->	Select Group	Select Channel
Select Telestra	E:Ch5->	Select Group	Select Channel
	F:Ch6->	Select Group	Select Channel
Select Telestra	G:Ch7->	Select Group	Select Channel
	H:Ch8->	Select Group	Select Channel


SaveTheWor
FalconEsco
rt
F22
PaxRiverLo
Telestra1
Observer
F22
AWACS
Cockpit
Instructor
F22ACU
Production
ACU

3.8. Helpers

Helpers are additional specialized tools within Project Manager that aid the user in building various types of Project elements.



Helpers include:

- Channels
- Communications Plans
- Math Plans
- Domains
- Sound Repositories
- Radios
- Hosts
- Test Plans
- Hit Plans
- Speech Recognition (SR) Plans

Each Helper produces Project elements that are added to the Project Layout. Elements are stored under their own respective folder for easy visibility and access. In this fashion, libraries of reusable elements are created for future use.

Helpers allow the developer to quickly build and manage complex simulations by creating reusable elements. The Helper tools assist to ensure consistency and interoperability within the simulation application.

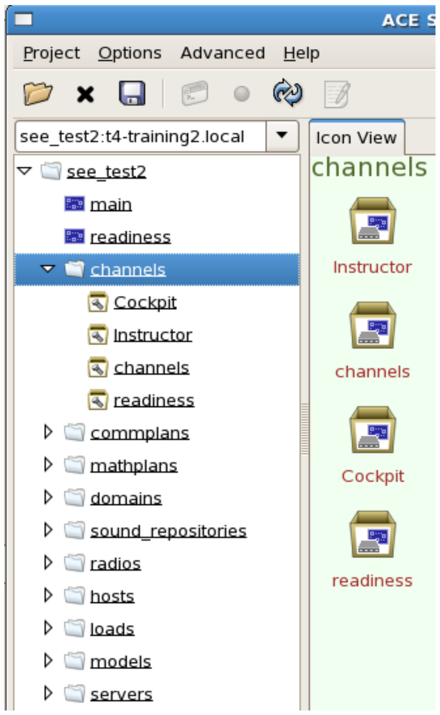
Note: When using the Helpers to create models, a level of detail is removed from the model building process. Developers cannot modify the helper-created model in the ACE Studio model builder, as all changes made at that level will be lost.

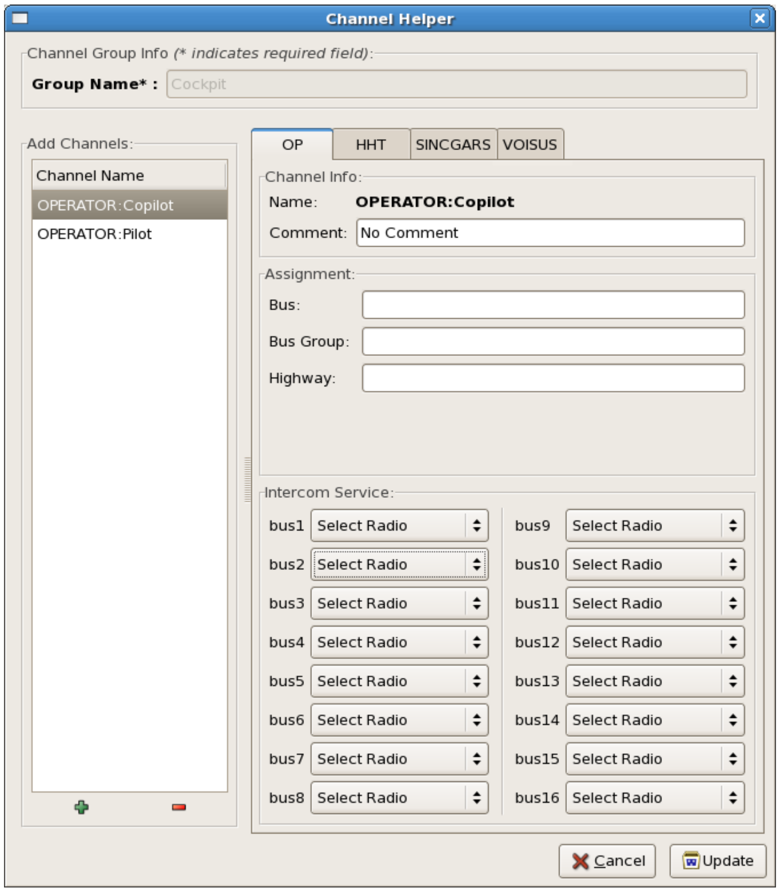
3.8.1. Channels

At their basic level, channels are audio connection points within the sound and communication model. A simple example is a mix of environmental cue sounds that must be routed to a particular speaker location. Another, more complex example is an operator that not only includes a microphone input and headset output but a communications panel structure and communications asset links (radios, intercoms, etc.).

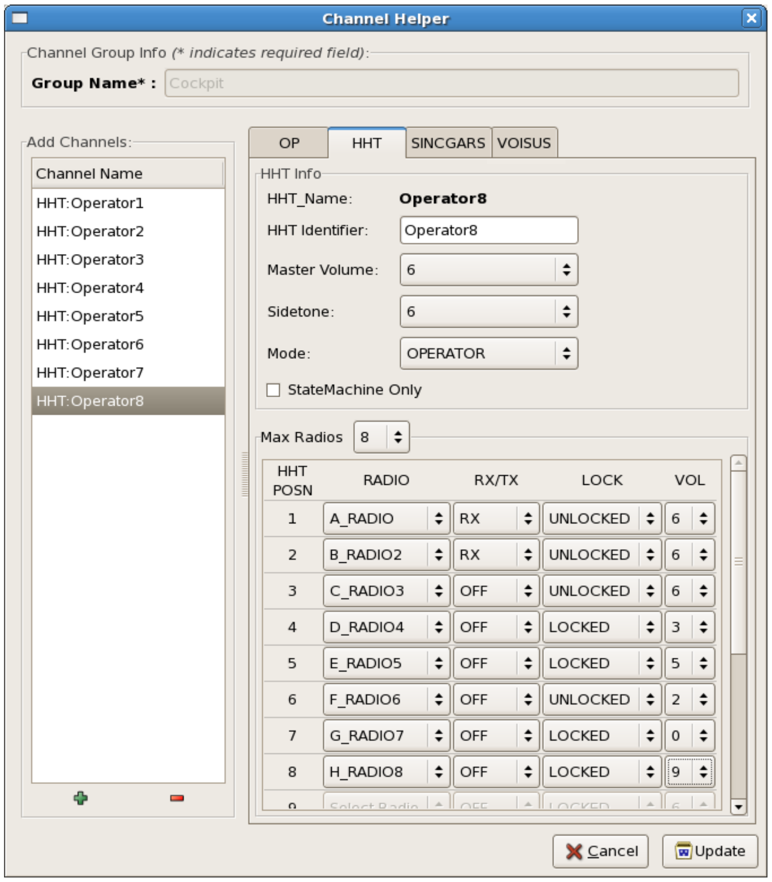
While the user can generate these structures using ACE Studio, the Channel Helper is a useful alternative as it can auto-generate some of these more complex audio related modeling structures. The Channel Helper creates modeling elements, which can be reused in a sound or communications model.

As an example, it is much easier and more consistent to use the Channel Helper to generate three (3) generic IOS operator positions than to manually generate these in ACE Studio's modeling environment.





In addition to adding generic operators, users can also add Hand-held Terminal (HHT) and SINCGARS Operators. Select the “+” button, add the operator and then click on the corresponding tab to manipulate the Operator position information.



Group Name* : Cockpit

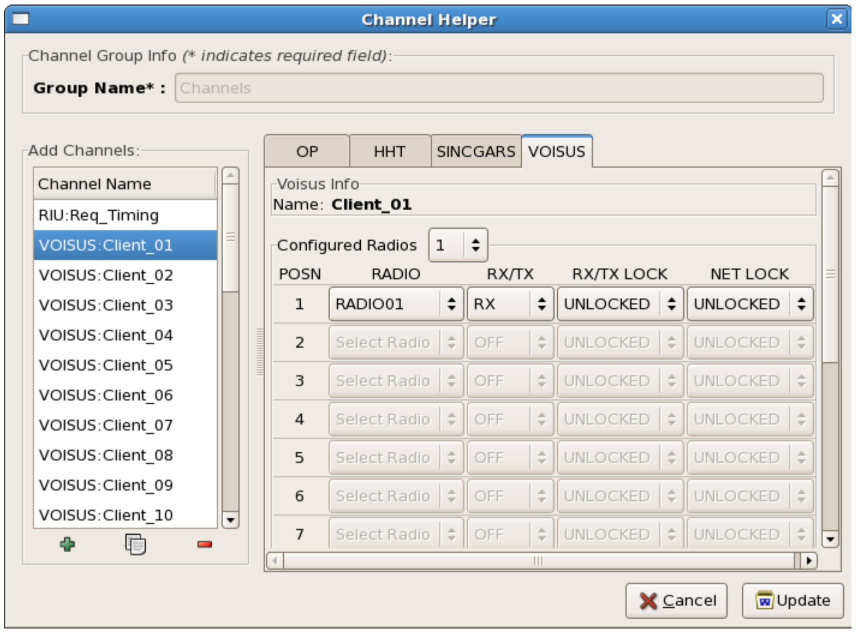
Channel Name
HHT:Operator1
HHT:Operator2
HHT:Operator3
HHT:Operator4
HHT:Operator5
HHT:Operator6
HHT:Operator7
HHT:Operator8

HHT_Name: **Operator8**
HHT Identifier: Operator8
Master Volume: 6
Sidetone: 6
Mode: OPERATOR
☐ StateMachine Only

Max Radios 8

HHT POSN	RADIO	RX/TX	LOCK	VOL
1	A_RADIO	RX	UNLOCKED	6
2	B_RADIO2	RX	UNLOCKED	6
3	C_RADIO3	OFF	UNLOCKED	6
4	D_RADIO4	OFF	LOCKED	3
5	E_RADIO5	OFF	LOCKED	5
6	F_RADIO6	OFF	UNLOCKED	2
7	G_RADIO7	OFF	LOCKED	0
8	H_RADIO8	OFF	LOCKED	9

The Voisus tab allows you to create Voisus Client operators. The Voisus Clients are hosted on the Synapse Voisus Server. ASTi's Voisus client-server technology offers an economical solution for network voice communications, simulation and audio distribution. See the Synapse Voisus Server Manual (DOC-01-SYN4-VS-1) for a complete system overview and client configuration.



Channel Helper

Channel Group Info (* indicates required field):

Group Name* : Channels

Add Channels:

Channel Name
RIU:Req_Timing
VOISUS:Client_01
VOISUS:Client_02
VOISUS:Client_03
VOISUS:Client_04
VOISUS:Client_05
VOISUS:Client_06
VOISUS:Client_07
VOISUS:Client_08
VOISUS:Client_09
VOISUS:Client_10

OP | HHT | SINCGARS | VOISUS

Voisus Info

Name: **Client_01**

Configured Radios 1

POSN	RADIO	RX/TX	RX/TX LOCK	NET LOCK
1	RADIO01	RX	UNLOCKED	UNLOCKED
2	Select Radio	OFF	UNLOCKED	UNLOCKED
3	Select Radio	OFF	UNLOCKED	UNLOCKED
4	Select Radio	OFF	UNLOCKED	UNLOCKED
5	Select Radio	OFF	UNLOCKED	UNLOCKED
6	Select Radio	OFF	UNLOCKED	UNLOCKED
7	Select Radio	OFF	UNLOCKED	UNLOCKED

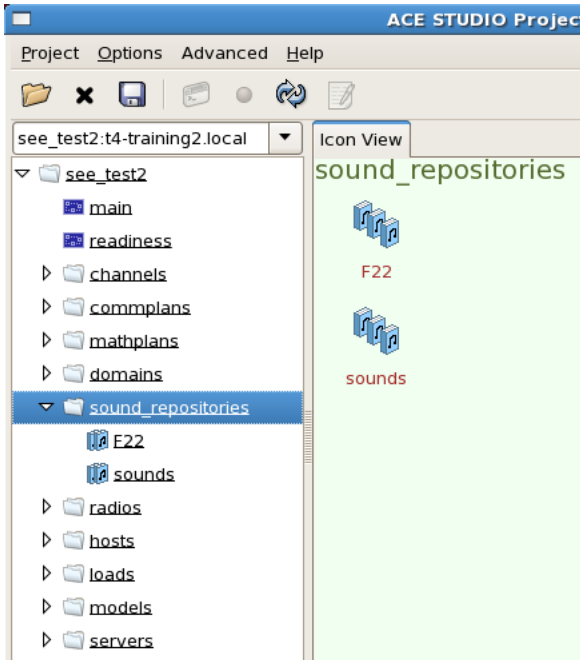
4.0. Model Services

4.1. Intercoms

Intercom components relate to internal communication paths within the model. This group includes the communication panel and local intercom buses. Audio on intercom buses is never transmitted onto the voice network. These buses are used internally to pass audio around. If an intercom is put in a radio, for example, the audio can be sent out on the DIS network.

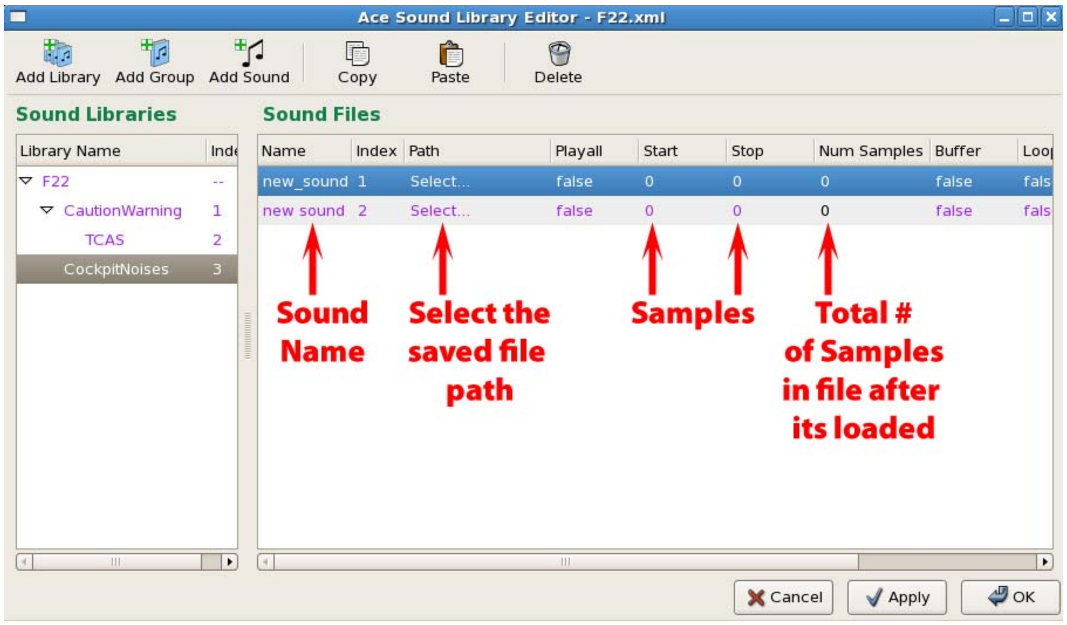
Intercoms provide an intercom audio bus structure to which other components can connect for distributing audio throughout a model and to simulate intercom bus structures in simulation applications. A network version allows an extension of intercom busses between systems using simulation industry standards DIS or HLA protocols.

4.2. Sound Repositories



Sound Repositories are essentially the sound file libraries needed to support user-developed sound and communication models. For example, an AH-64D Apache model may use a library of recorded sounds such as weapon launches, engine igniters, touchdown thump, caution/ warning voice alerts, threat alerts, crypto tones, FH equipment beeps and squawks, etc. as part of its simulation.

The Sound Library Editor allows the user to create, reuse, and manage these libraries. The editor also provides some convenient structuring of these sounds, allowing the user to build sound groups within an individual sound library. For example, a user might create an F18 communications sound library consisting of 20 or so recorded sound files. These sounds can then be grouped inside the library by function such as ARC182, Caution, Weapons, and Miscellaneous. The sound repository itself can be reused and applied to multiple F18 platforms.


Ace Sound Library Editor - F22.xml
Add Library
Add Group
Add Sound
Copy
Paste
Delete
Sound Libraries
Sound Files
Sound Name
Select the saved file path
Samples
Total # of Samples in file after its loaded
Cancel
Apply
OK

Library Name	Inde
F22	--
CautionWarning	1
TCAS	2
CockpitNoises	3

Name	Index	Path	Playall	Start	Stop	Num Samples	Buffer	Loo
new_sound	1	Select...	false	0	0	0	false	fals
new sound	2	Select...	false	0	0	0	false	fals

4.2.1. Creating a Sound Library

In the Project Manager > Sound Repositories, right-click to add a new sound library and name it accordingly. Open the Sound Library Editor and add groups, as necessary, to the library. The groups provide organization of the sound files.

To upload sound files to the Target open RMS and navigate to the Audio section. Open the Upload Sound Files page and follow the instructions to add individual WAV files or upload one TGZ archive containing multiple sound files. See the Remote Management System 4 User Guide (DOC-01-TEL4-RMS4-UG-4) for more information on uploading sound files.

Important: All sound files must be in the following format: 16-bit PCM MONO wave files with a 48khz sample rate. RMS will give you an error if the file is not in this format.

To locate sound files on the Target you must ‘ssh’ into the Target. From the ACE Studio Project Manager right-click the Telestra in the Layout and select ‘ssh.’

In the command line type the following:

```
cd var/local/asti
```

or

```
cd var/local/asti/soundfiles/<waveset>
```

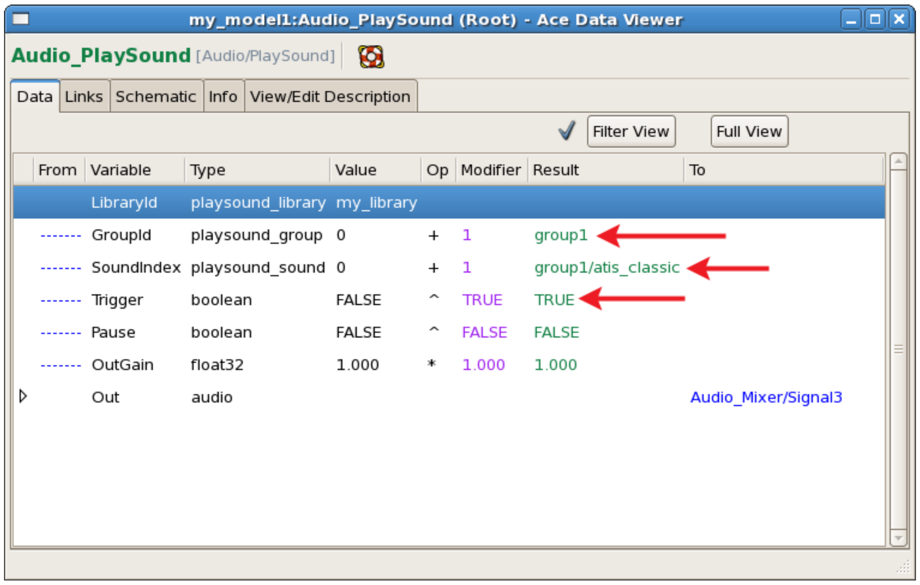
See **Appendix D: Useful Linux Commands** in the Target Operation and Maintenance Manual (DOC-01-TEL4-TUG-1) for more commands.

Note: The user must define the waveset in the 'Target Setup' in order to use the sound library. The waveset is the folder name that contains the wave files.

To create a simple sound object follow the steps below.

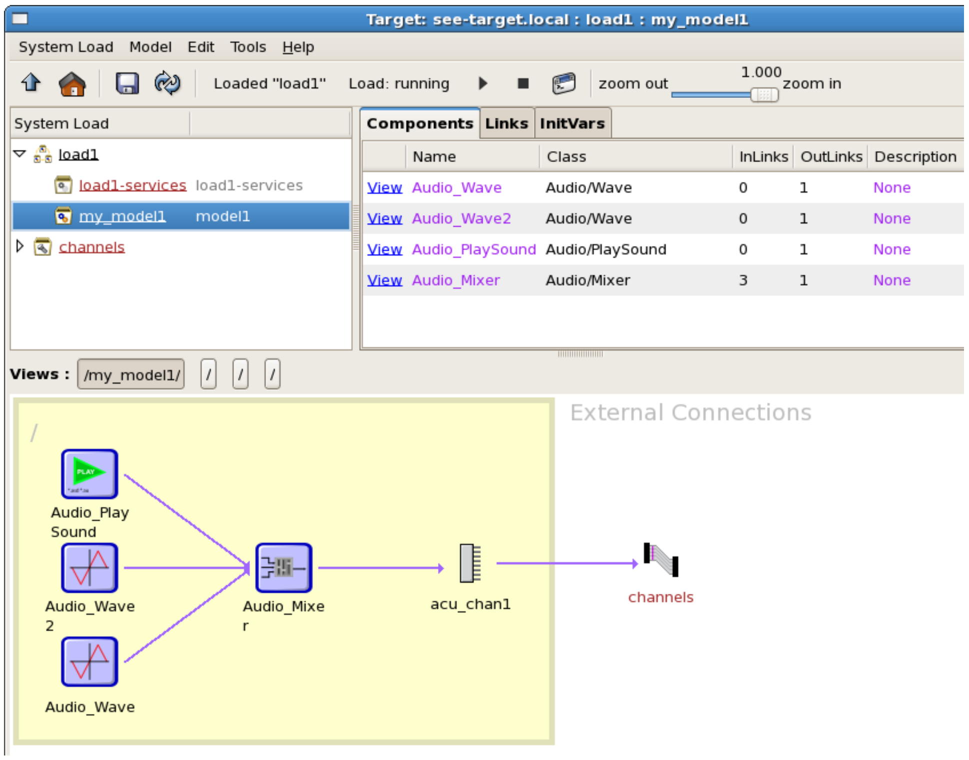
1. Create a model and add a playsound and mixer object.
2. Use the Link Editor to link the playsound to the mixer object.
3. In the playsound object select the sound library, group ID, and sound index.

The user should hear the sound file audio through the headset.


my_model1:Audio_PlaySound (Root) - Ace Data Viewer
Audio_PlaySound [Audio/PlaySound]
Data
Links
Schematic
Info
View/Edit Description
Filter View
Full View

From	Variable	Type	Value	Op	Modifier	Result	To
	LibraryId	playsound_library	my_library				
-------	GroupId	playsound_group	0	+	1	group1	
-------	SoundIndex	playsound_sound	0	+	1	group1/atis_classic	
-------	Trigger	boolean	FALSE	^	TRUE	TRUE	
-------	Pause	boolean	FALSE	^	FALSE	FALSE	
-------	OutGain	float32	1.000	*	1.000	1.000	
▷	Out	audio					Audio_Mixer/Signal3

Simple Sound Library Example Model

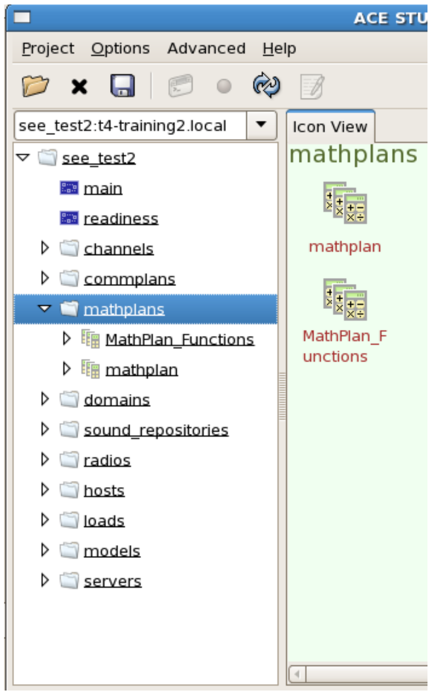


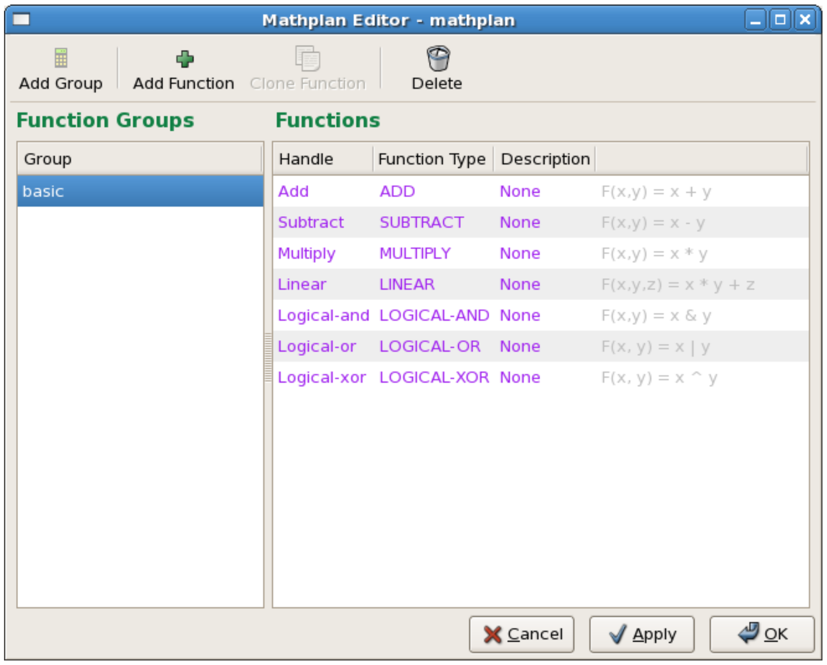
	Name	Class	InLinks	OutLinks	Description
View	Audio_Wave	Audio/Wave	0	1	None
View	Audio_Wave2	Audio/Wave	0	1	None
View	Audio_PlaySound	Audio/PlaySound	0	1	None
View	Audio_Mixer	Audio/Mixer	3	1	None

4.3. Math Plan

In Projects, the Math Plan provides an area to build and define groups of math functions. The Math Plan has a default set of math functions or a new math plan can be created with a blank canvas for creating new math functions.

The math functions are organized into 'Function Groups' for intended use in the model. For example, create a group for all engine functions. In the Math Plan, functions are added in generic form, i.e. they can be used by any component in the model. The functions do not hold specific information for a component. For example, the 'Add' handle is placed in two different components in a model. The inputs and outputs for this function are different in each component but it will perform the Add function for each set of variables.



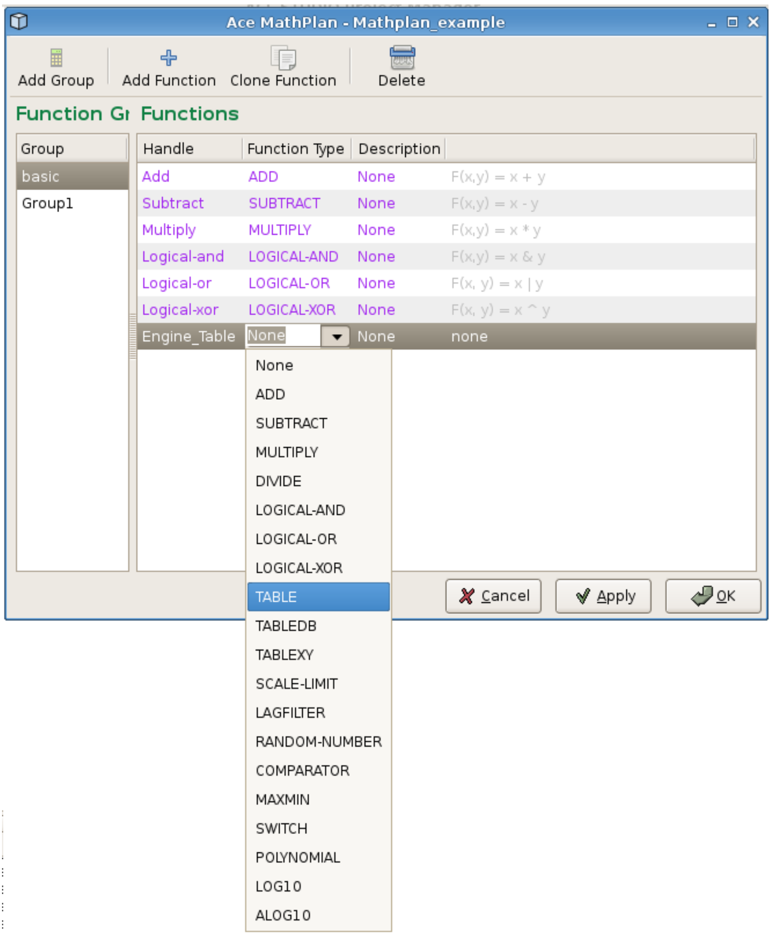


Group
basic

Handle	Function Type	Description	
Add	ADD	None	F(x,y) = x + y
Subtract	SUBTRACT	None	F(x,y) = x - y
Multiply	MULTIPLY	None	F(x,y) = x * y
Linear	LINEAR	None	F(x,y,z) = x * y + z
Logical-and	LOGICAL-AND	None	F(x,y) = x & y
Logical-or	LOGICAL-OR	None	F(x, y) = x | y
Logical-xor	LOGICAL-XOR	None	F(x, y) = x ^ y

All Math Plans have a basic set of standard math functions. The user can then add functions as needed. For each function, the ‘Handle’ is the name that describes the function. When the component points to the function in a model, it will use the Handle.

The ‘Function Type’ provides the drop down list of all the available functions. In the description field, add specific details about each function. The last column, which is not labeled, displays the type of configuration associated with each individual function.



Group
basic
Group1

Handle	Function Type	Description	
Add	ADD	None	F(x,y) = x + y
Subtract	SUBTRACT	None	F(x,y) = x - y
Multiply	MULTIPLY	None	F(x,y) = x * y
Logical-and	LOGICAL-AND	None	F(x,y) = x & y
Logical-or	LOGICAL-OR	None	F(x, y) = x | y
Logical-xor	LOGICAL-XOR	None	F(x, y) = x ^ y
Engine_Table	None	None	none

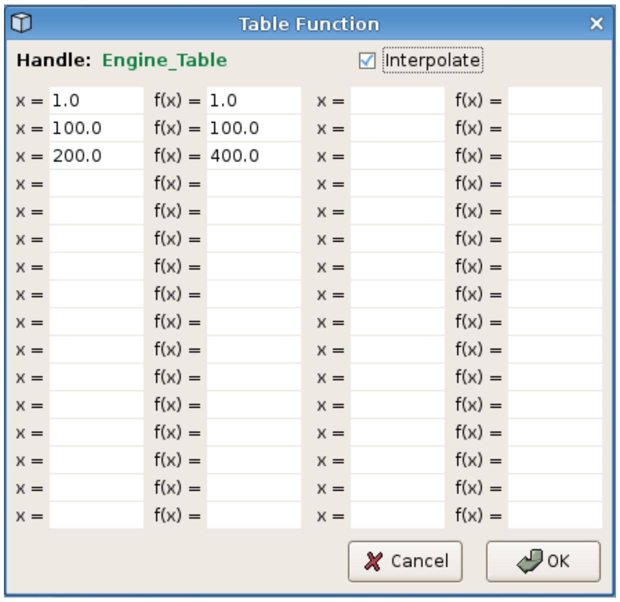
None
ADD
SUBTRACT
MULTIPLY
DIVIDE
LOGICAL-AND
LOGICAL-OR
LOGICAL-XOR
TABLE
TABLEDB
TABLEXY
SCALE-LIMIT
LAGFILTER
RANDOM-NUMBER
COMPARATOR
MAXMIN
SWITCH
POLYNOMIAL
LOG10
ALOG10

4.3.1. Math Functions

The math functions permit local manipulation of data within the models. The remainder of this section describes each math function in detail.

Many of the math functions have the option of interpolation. An interpolation parameter allows for either a non-interpolated stepwise output based upon the closest value to the input, or a linear interpolation of data between adjacent table inputs.

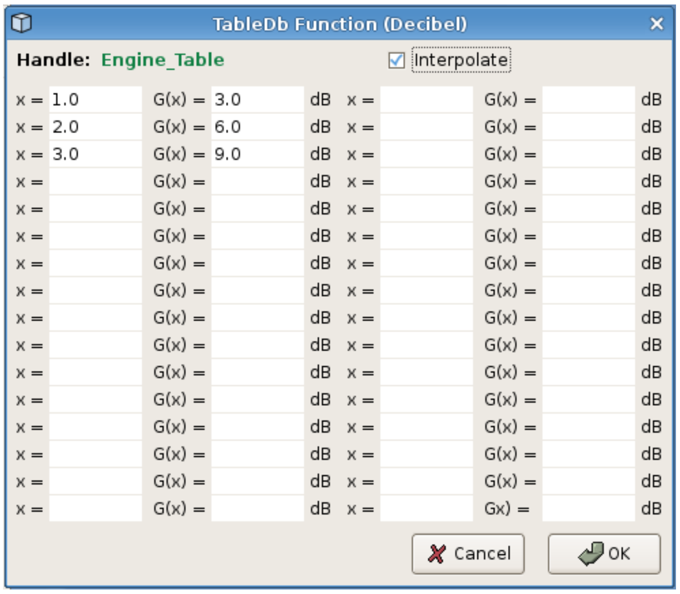
Table – Provides a look-up table for definable values.



Handle: Engine_Table   [✓] Interpolate

x =	1.0	f(x) =	1.0	x =		f(x) =	
x =	100.0	f(x) =	100.0	x =		f(x) =	
x =	200.0	f(x) =	400.0	x =		f(x) =	
x =		f(x) =		x =		f(x) =	
x =		f(x) =		x =		f(x) =	
x =		f(x) =		x =		f(x) =	
x =		f(x) =		x =		f(x) =	
x =		f(x) =		x =		f(x) =	
x =		f(x) =		x =		f(x) =	
x =		f(x) =		x =		f(x) =	
x =		f(x) =		x =		f(x) =	
x =		f(x) =		x =		f(x) =	
x =		f(x) =		x =		f(x) =	
x =		f(x) =		x =		f(x) =	
x =		f(x) =		x =		f(x) =	
x =		f(x) =		x =		f(x) =	

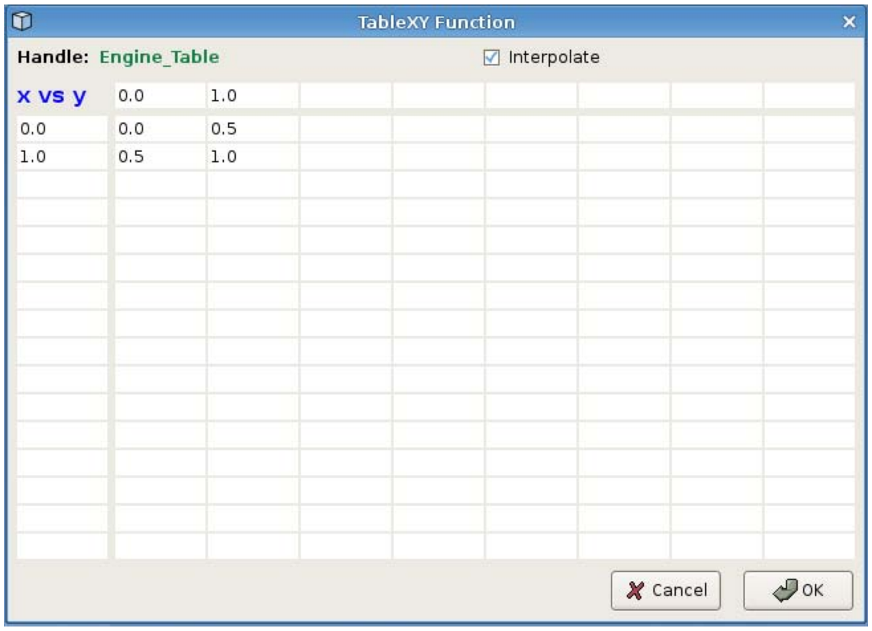
Table dB – Provides the same as ‘Table’ except the output is in decibels (dB).



Handle: Engine_Table   [✓] Interpolate

x =	1.0	G(x) =	3.0	dB	x =		G(x) =		dB
x =	2.0	G(x) =	6.0	dB	x =		G(x) =		dB
x =	3.0	G(x) =	9.0	dB	x =		G(x) =		dB
x =		G(x) =		dB	x =		G(x) =		dB
x =		G(x) =		dB	x =		G(x) =		dB
x =		G(x) =		dB	x =		G(x) =		dB
x =		G(x) =		dB	x =		G(x) =		dB
x =		G(x) =		dB	x =		G(x) =		dB
x =		G(x) =		dB	x =		G(x) =		dB
x =		G(x) =		dB	x =		G(x) =		dB
x =		G(x) =		dB	x =		G(x) =		dB
x =		G(x) =		dB	x =		G(x) =		dB
x =		G(x) =		dB	x =		G(x) =		dB
x =		G(x) =		dB	x =		G(x) =		dB
x =		G(x) =		dB	x =		G(x) =		dB
x =		G(x) =		dB	x =		Gx) =		dB

Table XY – X verses Y provides a two dimensional table that looks up the intersection of the X and Y. The intersection is the output.

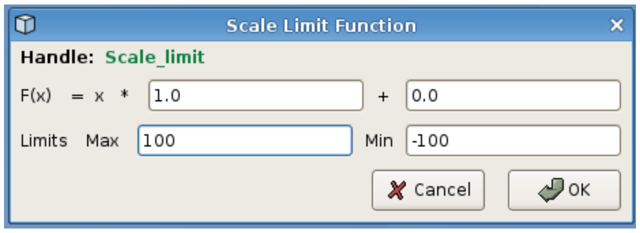


Handle: Engine_Table

Interpolate ✓

x vs y	0.0	1.0
0.0	0.0	0.5
1.0	0.5	1.0

Scale Limit – Applies a maximum or minimum value to 'X'. If the value is over the maximum or under the minimum the limit values are used. The output can also be scaled and offset.



Handle: Scale_limit

F(x) = x * 1.0 + 0.0

Limits Max 100 Min -100

Lagfilter – The lag filter provides a simple slew-rate limiting filter, which is useful for fade-in and fade-out effects. The filter can be used to smooth harsh transitions within a model and cause a quick numerical jump to change slowly over time.

The filter function is defined as:

Y N=Y N-1+ K (X N- Y N-1)

Where:

X N = new input value

Y N = new output value

Y N-1 = last frame's output value

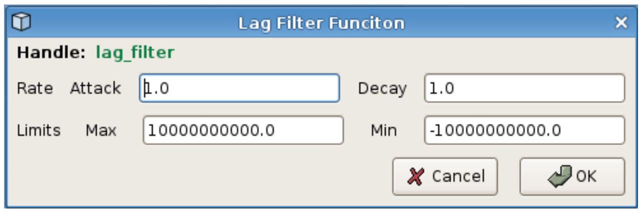
K = Attack const If X N > Y N-1

K = Decay const If X N < Y N-1

The **Attack rate** determines how quickly the function goes from the old value to the new value when the new value is greater than the old value. The smaller the attack rate the slower the function (or the longer the lag).

The **Decay rate** determines how quickly the function goes from the old value to the new value when the new value is less than the old value. The smaller the attack rate the slower the function (or the longer the lag).

The maximum limit value and minimum limit value are for filter accumulation to prevent saturation of integrators.

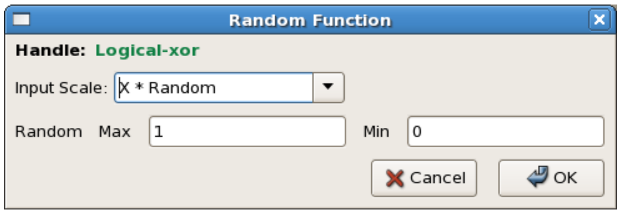


Handle: lag_filter

Rate Attack 1.0 Decay 1.0

Limits Max 10000000000.0 Min -10000000000.0

Random Function – The random function generates a number between 0.0 and 1.0. The user selects to add or multiply the input value by the random number. The output can also be scaled and offset. The maximum and minimum numbers limit the output value within the desired range.

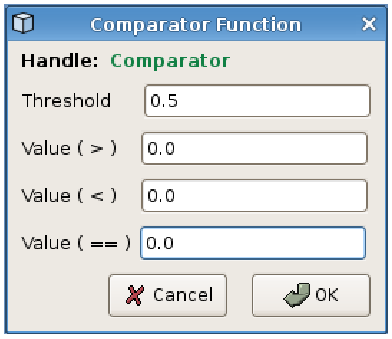


Handle: Logical-xor

Input Scale: X * Random

Random Max 1 Min 0

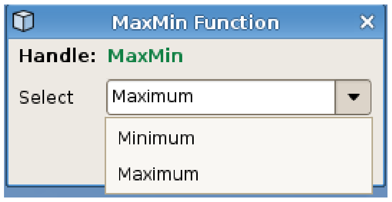
Comparator – The comparator function consists of two inputs, X and Y. X is compared against the value of Y. X can either be equal to, less than, or greater than Y. Based upon the result of the comparison between X and Y, the output will be one of the corresponding user defined values for each of the three possible results.



Handle: Comparator

Threshold	0.5
Value (>)	0.0
Value (<)	0.0
Value (==)	0.0

MaxMin – Select the maximum or minimum between two values.

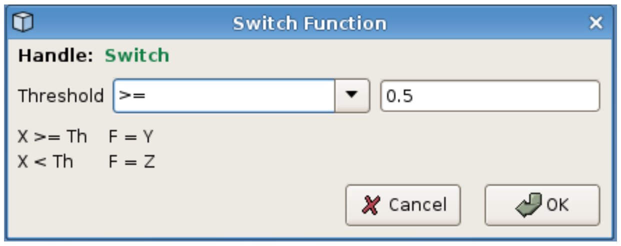


Handle: MaxMin

Select: Maximum

Switch – The switch component provides the ability to switch between two values dependent upon a conditional threshold. There are three input values, X, Y and Z. Y and Z are the output values and X is the value upon which the condition is tested. If the condition is TRUE, then Y is the output otherwise Z is the output. The functionality of the component can be read like so:

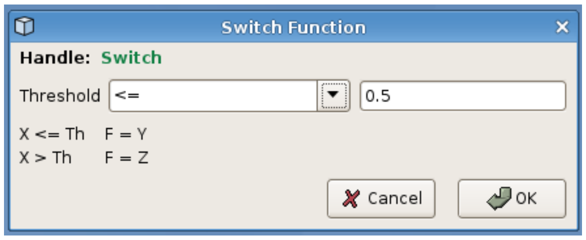
```
If ‘X> =,’ or If ‘X < = '  threshold then
  Output Y
Else
  Output Z
endif
```


Switch Function

Handle: Switch

Threshold >= 0.5

X >= Th F = Y
X < Th F = Z

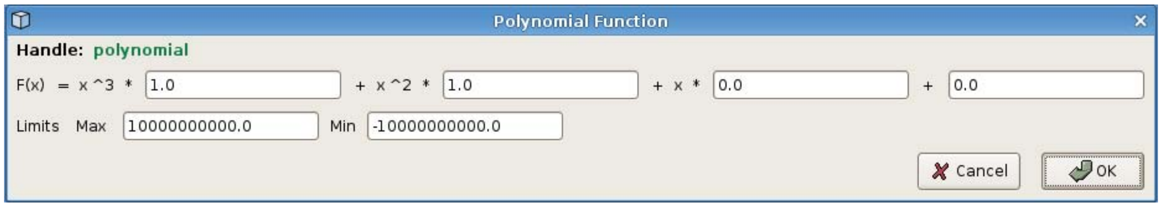

Switch Function

Handle: Switch

Threshold <= 0.5

X <= Th F = Y
X > Th F = Z

Polynomial – Allows the user to specify up to a third order polynomial. The maximum and minimum limits the output value within the desired range.

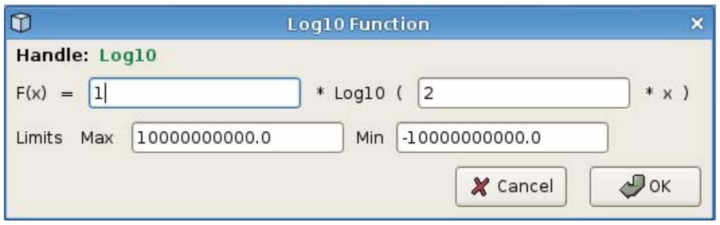

Polynomial Function

Handle: polynomial

F(x) = x ^3 * 1.0 + x ^2 * 1.0 + x * 0.0 + 0.0

Limits Max 10000000000.0 Min -10000000000.0

Log10 – Specifies the log base 10 function. The maximum and minimum limits the output value within the desired range.

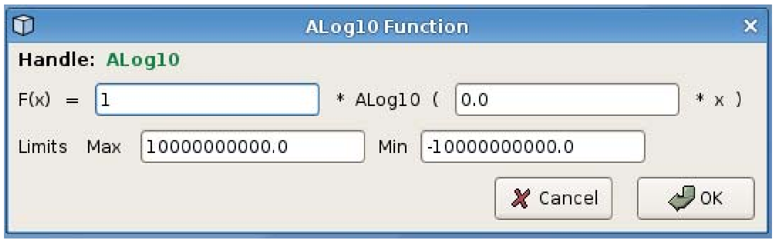


Handle: Log10

F(x) = 1 * Log10 (2 * x)

Limits Max 10000000000.0 Min -10000000000.0

Alog10 – Specifies the inverse of log base 10. The maximum and minimum limits the output value within the desired range.



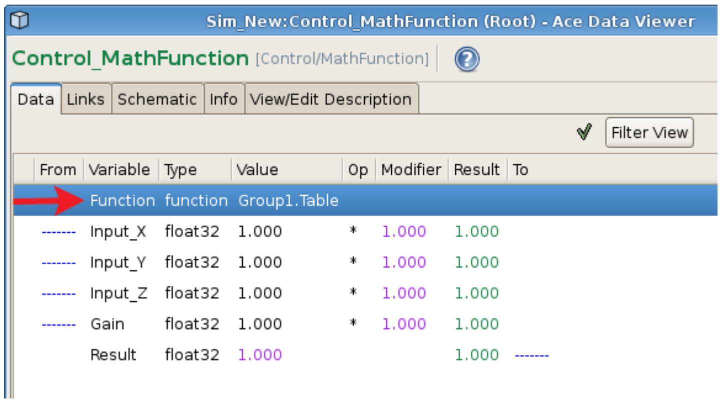
Handle: ALog10

F(x) = 1 * ALog10 (0.0 * x)

Limits Max 10000000000.0 Min -10000000000.0

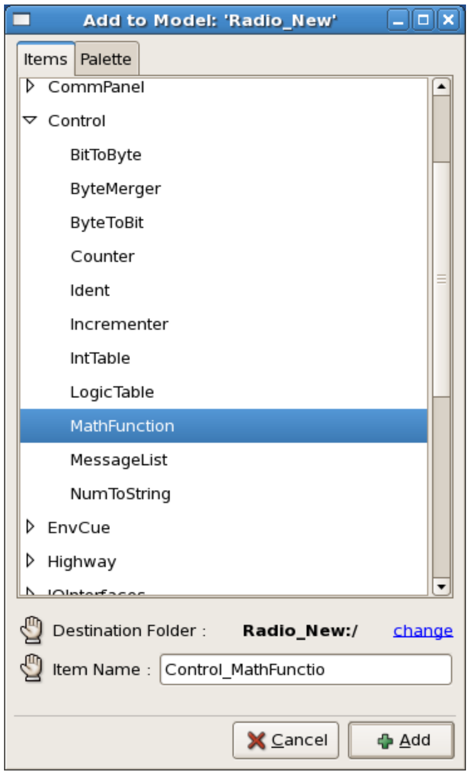
4.3.2. Where to use the Math Plan Functions

In ACE Studio model development, the math functions can only be used in certain components. The variable and the type must have ‘Function’ listed in the component data.


Sim_New:Control_MathFunction (Root) - Ace Data Viewer
Control_MathFunction [Control/MathFunction]
Data
Links
Schematic
Info
View/Edit Description
Filter View

From	Variable	Type	Value	Op	Modifier	Result	To
	Function	function	Group1.Table				
-------	Input_X	float32	1.000	*	1.000	1.000	
-------	Input_Y	float32	1.000	*	1.000	1.000	
-------	Input_Z	float32	1.000	*	1.000	1.000	
-------	Gain	float32	1.000	*	1.000	1.000	
	Result	float32	1.000			1.000	-------

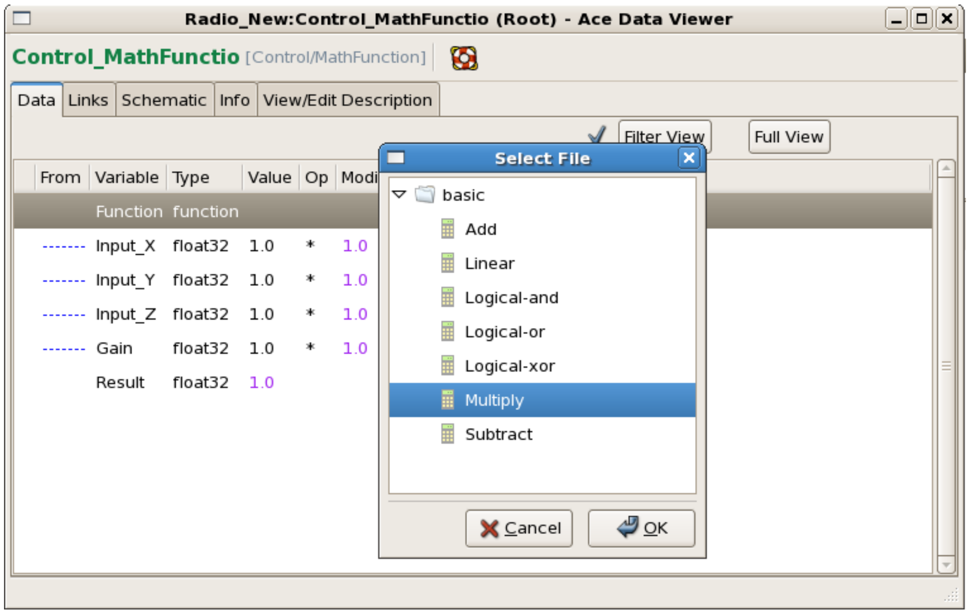
The Math Plan functions are most commonly used in the Math Function component.



Destination Folder : **Radio_New:/** change

Item Name : Control_MathFunctio

If the function is a single variable function it will use the X input, for two variables it will use the X and Y inputs, for three variables it will use the X, Y, and Z inputs.


Radio_New:Control_MathFunctio (Root) - Ace Data Viewer
Control_MathFunctio [Control/MathFunction]
Data
Links
Schematic
Info
View/Edit Description
Filter View
Full View
Select File
Cancel
OK

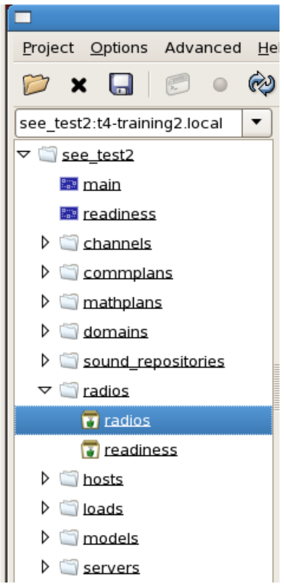
From	Variable	Type	Value	Op	Modi
	Function	function			
-------	Input_X	float32	1.0	*	1.0
-------	Input_Y	float32	1.0	*	1.0
-------	Input_Z	float32	1.0	*	1.0
-------	Gain	float32	1.0	*	1.0
	Result	float32	1.0		

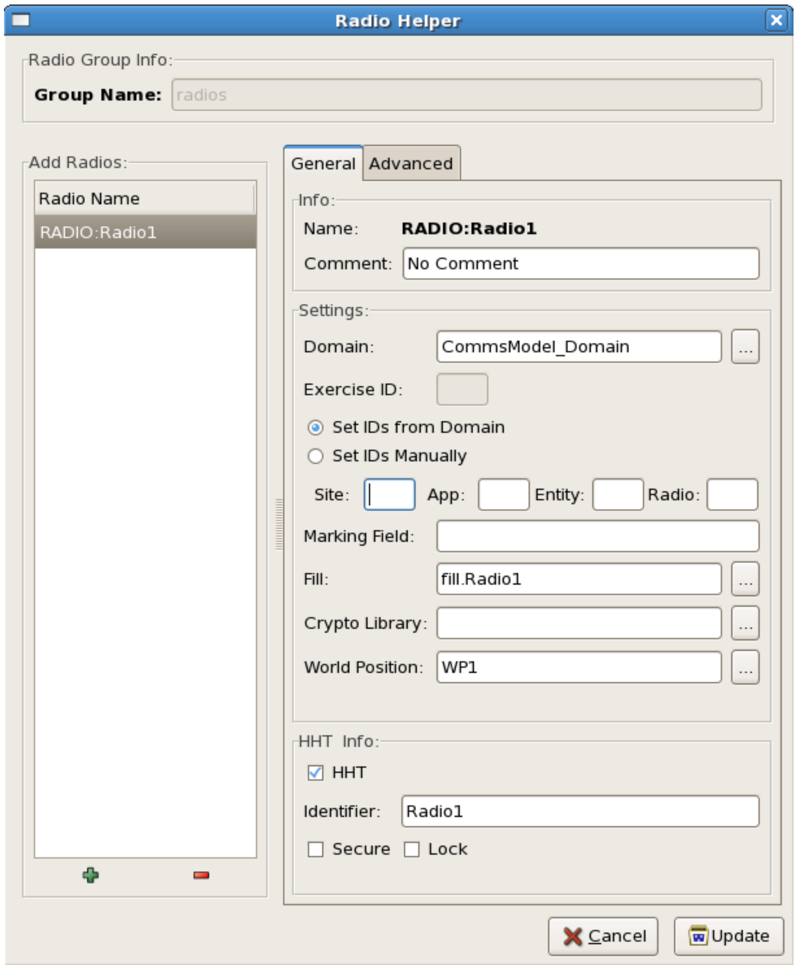
- basic
  - Add
  - Linear
  - Logical-and
  - Logical-or
  - Logical-xor
  - Multiply
  - Subtract

4.4. Radio Overview

As the name suggests, radios are the simulated radio assets installed by a developer when building a communications model.

Simulated radio components can be installed and used directly within the ACE Studio model development environment or the user can use the Radio Helper to auto-build radio simulation sections of their communications models.





Group Name: radios

RADIO:Radio1

Name: **RADIO:Radio1**

Comment: No Comment

Domain: CommsModel_Domain

Exercise ID:

(●) Set IDs from Domain

( ) Set IDs Manually

Site: App: Entity: Radio:

Marking Field:

Fill: fill.Radio1

Crypto Library:

World Position: WP1

☑ HHT

Identifier: Radio1

☐ Secure ☐ Lock

4.4.1. Simulated Radio Features

Radios are the largest, most complex, and most utilized components in ACE Studio. The following is a summary list of radio features for all simulated radios.

- **World Position –** the x, y, z coordinates of the radio's location. The radio must be connected to a world position, so it has a position to perform ranging calculations with and to identify the radio in the simulated environment.
- **Frequency –** defines the center of the radio tune frequency for transmit and receive. Optionally, it can define separate transmit and receive frequencies. This must be set to a non-zero value for radio communication.
- **Antenna Gain –** this field simulates the size and radiative efficiency of the antenna. Note that all modeled antennae are isotropic.
- **Squelch –** a noise gate that only allows signals with a specified strength to filter through and play.
- **Background Noise –** general noise created when using radios.
- **Fill –** Allows the user to select one of a set of N pre-defined radio fills as defined in a global comm plan.
- **Multiple Net Support –** provides the radio with the ability to support multiple nets per a specific fill. Nets define the following core radio characteristics:
  - **Multiple Modulation Type –** describes the modulation parameters of the radio such as AM, FM, SATCOM, HQ, Intercom, etc.
  - **Amplitude Modulation (AM) and Frequency Modulation (FM) –** two primary modulations for radio operation.
  - **Modulation Discrimination –** occurs when radios can only receive signals from radios with the same modulation type.
  - **AM Mixing –** when multiple signals broadcast on the same channel frequency.
  - **FM Capture Effect –** when several FM radios are transmitting on the same frequency, an FM receiver will only be able to receive the strongest signal.
  - **Sensitivity –** receiver sensitivity in dB.
  - **Bandwidth and Bandwidth Overlap –** the bandwidth is used to determine the amount of audio noise mixed into the received audio, based on the simulated bandwidth of the radio band. This parameter does not affect the "in tune" calculation.
  - **Encoding Type and Rate –** defines audio encoding type (muLaw, CVSD, PCM) and the sample rate.
  - **Transmit Power –** indicates the transmission power of the radio (Watts / dBm)
  - **Automatic Gain Control (AGC) –** the adjusting of the gain to appropriate levels for a range of input signal levels.

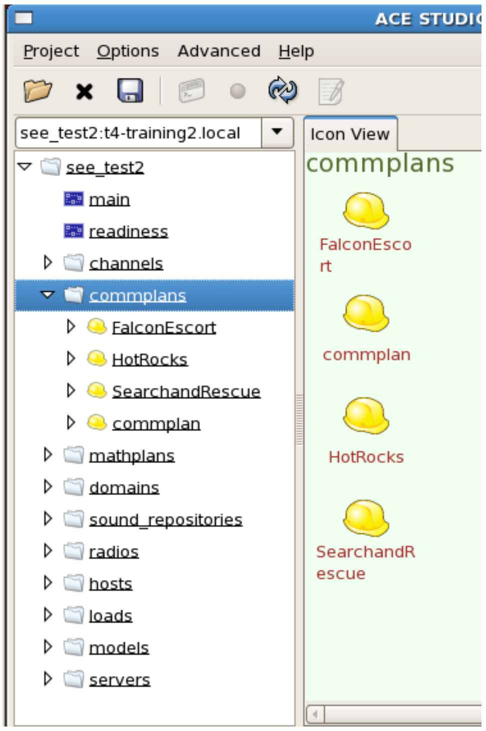
  - **SATCOM Parameters –** defines satellite mode parameters if applicable.
  - **Frequency Hopping –** a method of rapidly switching frequencies while a receiver and transmitter communicate. The receiver and transmitter have to jump between the same frequencies, at the same speed, and at the same time.
  - **Crypto Parameters –** radios that scramble the signals before they are transmitted so that only receivers who know the special key will have the ability to decode them, producing a secure voice transmission across any frequency.
- **Half Duplex and Full Duplex –** In half-duplex mode the radio is able to transmit and receive signals but cannot do both at the same time. Full-duplex allows radios to transmit and receive signals at the same time. Typically, full duplex is only used for intercom systems.
- **Propagation –** the movement of the radio waves as they transmit.
  - **Ranging –** an effect that occurs as a result of the distance between two radios. The greater the distance between the radios, the weaker the signal due to the dissipating power of the signal as it traverses a large area.
  - **Occulting –** the loss of radio signal due to the curvature of the earth’s horizon.
  - **Ionosphere Effects –** the loss of signal due to the changes in the earth’s atmosphere such as time of day or different seasons. The ionosphere effects only occur with High-Frequency (HF) radios.
  - **Line of Sight (LOS) –** when radio waves traveling in a straight line are dispersed due to obstacles or obstructions.
  - **Fresnel Diffraction –** loss of signal due to the reflection off obstacles in the path of the radio waves from transmitter to receiver.
- **Terrain Effects –** the loss of signal due to land obstruction such as a mountain.

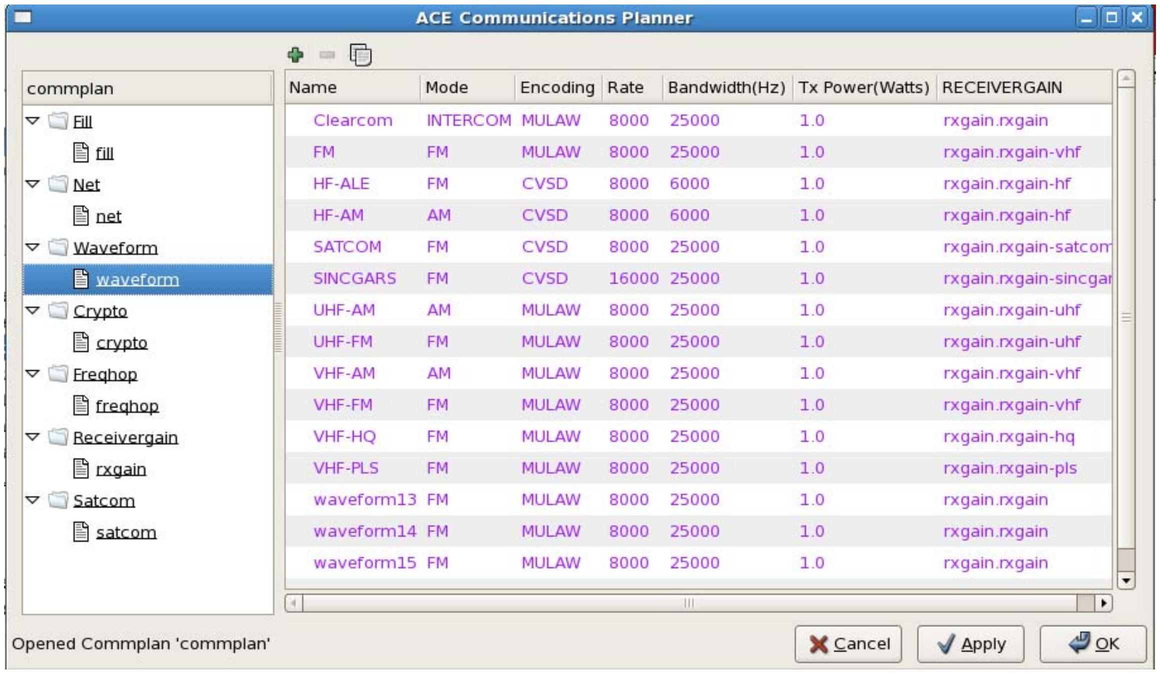
4.5. Communication Plans Overview

Much like the real world, a Communication Plan provides the necessary radio asset configuration parameters (or fills) for the radio assets playing in the simulated environment.

Using the Communication Plan (Commplan) tool, the user creates a library of radio “fills” consisting of crypto, frequency hop, waveform types, nets and other necessary parameters for the simulated radios in Project.

The user can create multiple communications plans and store them as part of the Project. In this way, different plans can be applied and executed with relative ease to support changing operational or exercise requirements. For example, the day-to-day operations of an F-16 simulator may utilize one plan, which provides the trainer communications simulation as tested and signed off with the device. However, other plans may be applied when the F-16 device is used in the network wide exercises.





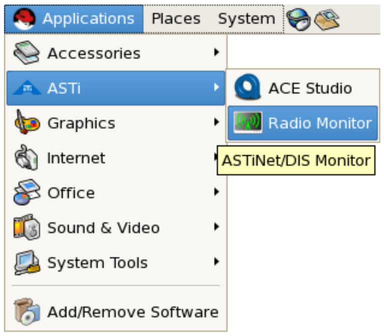
Name	Mode	Encoding	Rate	Bandwidth(Hz)	Tx Power(Watts)	RECEIVERGAIN
Clearcom	INTERCOM	MULAW	8000	25000	1.0	rxgain.rxgain
FM	FM	MULAW	8000	25000	1.0	rxgain.rxgain-vhf
HF-ALE	FM	CVSD	8000	6000	1.0	rxgain.rxgain-hf
HF-AM	AM	CVSD	8000	6000	1.0	rxgain.rxgain-hf
SATCOM	FM	CVSD	8000	25000	1.0	rxgain.rxgain-satcom
SINCGARS	FM	CVSD	16000	25000	1.0	rxgain.rxgain-sincgar
UHF-AM	AM	MULAW	8000	25000	1.0	rxgain.rxgain-uhf
UHF-FM	FM	MULAW	8000	25000	1.0	rxgain.rxgain-uhf
VHF-AM	AM	MULAW	8000	25000	1.0	rxgain.rxgain-vhf
VHF-FM	FM	MULAW	8000	25000	1.0	rxgain.rxgain-vhf
VHF-HQ	FM	MULAW	8000	25000	1.0	rxgain.rxgain-hq
VHF-PLS	FM	MULAW	8000	25000	1.0	rxgain.rxgain-pls
waveform13	FM	MULAW	8000	25000	1.0	rxgain.rxgain
waveform14	FM	MULAW	8000	25000	1.0	rxgain.rxgain
waveform15	FM	MULAW	8000	25000	1.0	rxgain.rxgain

Opened Commplan 'commplan'

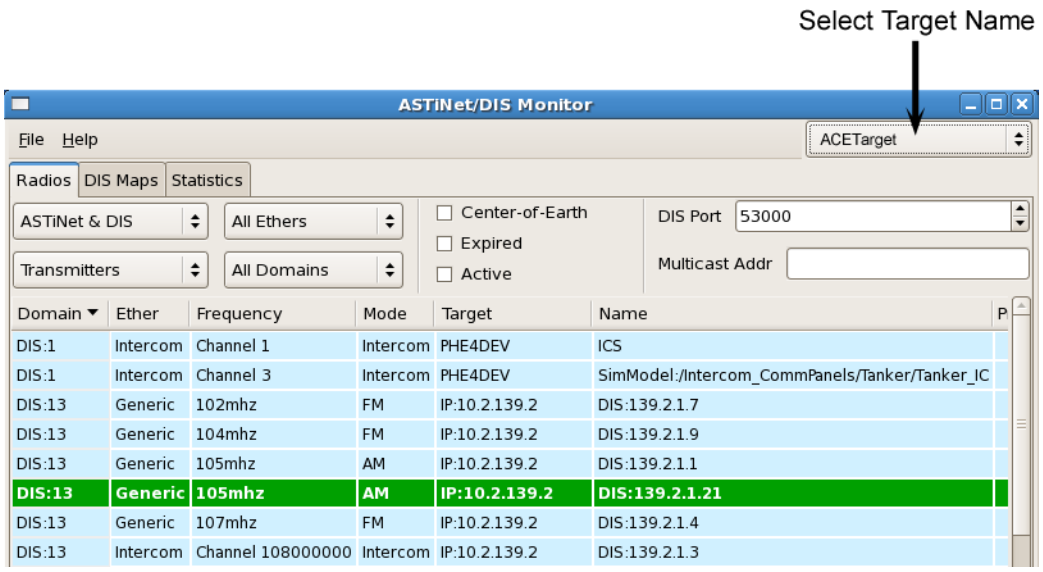
4.6. Radio Monitor

The Radio Monitor is a network-debugging tool that allows the user to examine the radio environment and other servers. The Radio Monitor updates in real time and can be accessed without a web browser. The radio monitor provides a view of the radio receivers on the network. The details include source or IP address, Name, Frequency, Bandwidth, Power (db), Audio Intune, Audio Counter, TDL Intune, Range, DARN ID and DIS ID.

To open Radio Monitor navigate to Applications > ASTi > Radio Monitor.



Select the Target name to view the radios on that system.



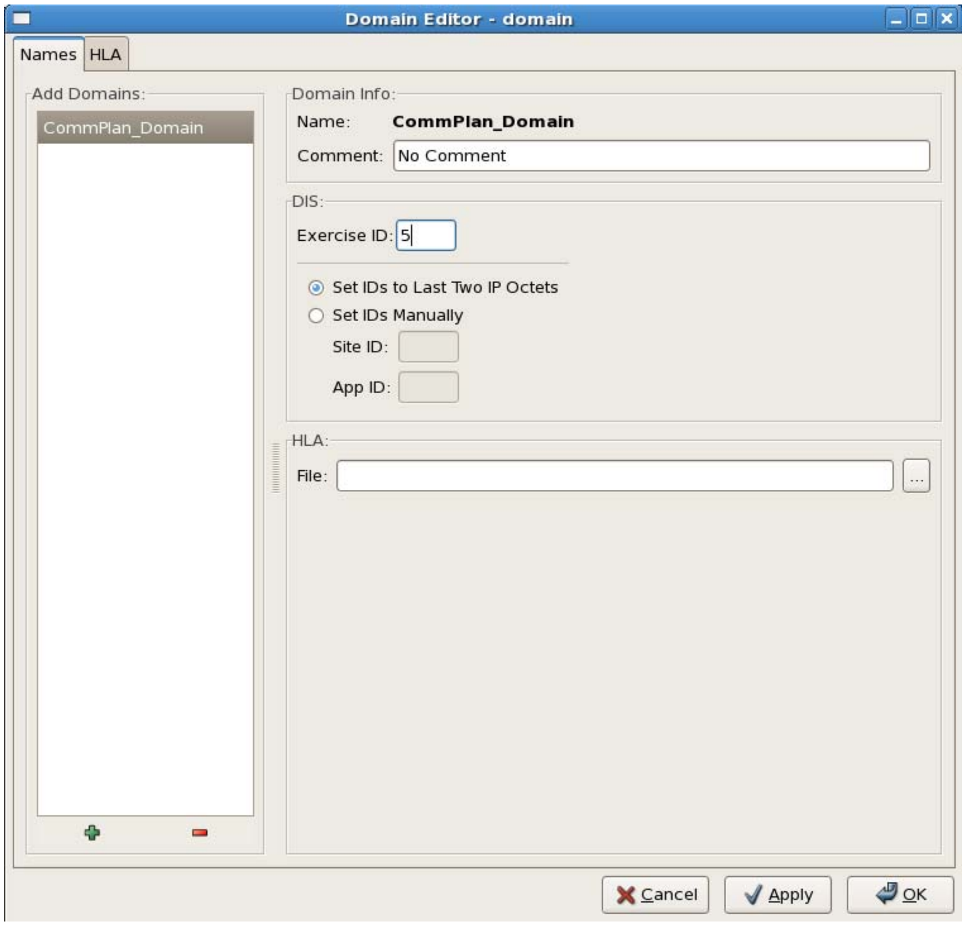
Domain	Ether	Frequency	Mode	Target	Name	P
DIS:1	Intercom	Channel 1	Intercom	PHE4DEV	ICS	
DIS:1	Intercom	Channel 3	Intercom	PHE4DEV	SimModel:/Intercom_CommPanels/Tanker/Tanker_IC	
DIS:13	Generic	102mhz	FM	IP:10.2.139.2	DIS:139.2.1.7	
DIS:13	Generic	104mhz	FM	IP:10.2.139.2	DIS:139.2.1.9	
DIS:13	Generic	105mhz	AM	IP:10.2.139.2	DIS:139.2.1.1	
DIS:13	**Generic**	**105mhz**	**AM**	**IP:10.2.139.2**	**DIS:139.2.1.21**	
DIS:13	Generic	107mhz	FM	IP:10.2.139.2	DIS:139.2.1.4	
DIS:13	Intercom	Channel 108000000	Intercom	IP:10.2.139.2	DIS:139.2.1.3	

4.7. Domain Editor

The Domain Editor provides the ability to set standard DIS or HLA parameters to apply to the project for a specific exercise. In a DIS exercise, each simulated radio must be uniquely identified by a combination of Exercise ID, Site ID, Application ID, Entity ID and Radio ID. Simulated radios must operate on the same Exercise ID with the same frequency to communicate with each other.

First, select the ‘+’ symbol to add a Domain and then add the exercise ID. Next fill in the required DIS parameters including Host site ID and App ID.

Important: Do not use the string “DIS:” in the Domain name as this is a command reserved for use in the software and the Domain will not work properly with this name. If sending host data that overwrites the Domain, do not use the same Exercise ID previously set in the Domain as DIS communication will not work properly.



CommPlan_Domain

Name: CommPlan_Domain

Comment: No Comment

Exercise ID: 5

[● Set IDs to Last Two IP Octets]

○ Set IDs Manually

Site ID:

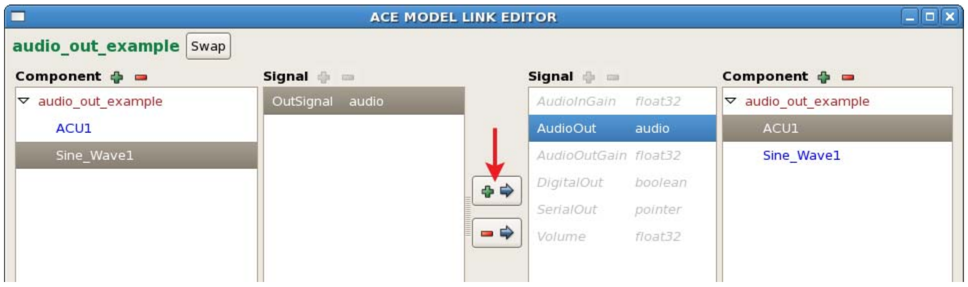
App ID:

File:

4.8. ACE Model Link Editor

To link the signals to and from model components, the user simply has to point and click. Select a component and a signal to route from the component. Then select the component and signal to route the signal. For the example below, the Sine_Wave1 OutSignal is routed to ACU1 AudioOut.

Note: Selecting the components first will display the signal options.



Component	Signal		Signal		Component
audio_out_example	OutSignal	audio	AudioInGain	float32	audio_out_example
ACU1			AudioOut	audio	ACU1
Sine_Wave1			AudioOutGain	float32	Sine_Wave1
			DigitalOut	boolean	
			SerialOut	pointer	
			Volume	float32	

4.9. Highway 3D Service

Audio Routing and 3D Soundfield Reconstruction

This service is a collection of components used for mixing and routing audio to hardware output channels. The Highway 3D Service provides two approaches for audio mixing - Gain Mixing and 3D Soundfield Reconstruction - making it useful for a range of applications, from routing communications audio to operator headsets to immersing listeners in a 3D audio environment.

See the ACE Studio Component Reference Guide (DOC-01-TELAS-CRG-4) “Highway 3D Service” section for more information.

4.9.1. Setup

To start using the service, link component audio outputs to the input of an Audio/AudioFeed component. In the AudioFeed, create a “Cue ID” for the audio stream. The Cue ID is a name that is used to reference the audio stream in other Highways 3D service components.

4.9.2. Two Approaches

The Highways 3D service processes the named audio stream from the AudioFeed component in two different ways, depending on which type of Feeder component is used - a Balancer or an AuralCuePos.

1. **Standard Gain-Mixing** with a Balancer component provides simple mixing of the named audio stream to a set of output “highway” audio streams. Balancers come in various sizes (Balancer1, Balancer4, etc.), with the number indicating the maximum number of highways to mix to. This approach is similar to highway use in the DACS.
2. **3D Soundfield Reconstruction** with an AuralCuePos component encodes the named audio stream into a 3D soundfield that is listened to with headphones or a speaker array. A 3D position specified in the AuralCuePos component tells the service where to localize the audio in the soundfield.

4.9.3. Standard Gain-Mixing

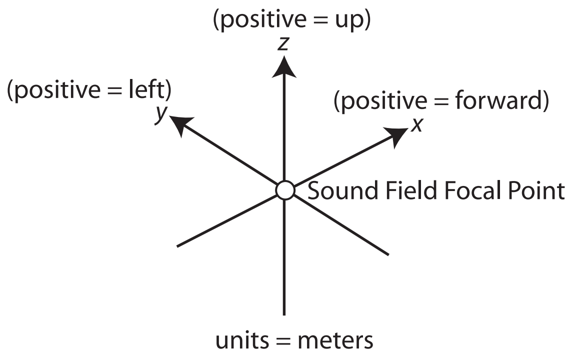
Use this approach for simple audio mixing to a small number of output channels. For example, use a Balancer to mix a tone to a headset and speaker, with the tone twice as loud in the speaker than in the headphones. In this example, two output highways are used, one for the headset and one for the speaker.

In the Balancer component, select a named audio stream to mix, select the highways to mix to, and then set the gains that are used when mixing the audio to each highway. Next, add an AudioIO/HighwayOut component to extract audio from each highway. This highway audio is then typically linked to an ACE-RIU or ACU audio output.

4.9.4. 3D Soundfield Reconstruction

Use this approach for routing 3D audio through headphones or a speaker array. For example, route helicopter rotor audio from a PlaySound component sound as if it was overhead by entering in an appropriate 3D position in the AuralCuePos component.

In the AuralCuePos component, select a named audio stream to mix and enter or link to the position values (X, Y, and Z). The position is specified in meters, relative to the listening or “reference” point. The orientation of the coordinate system is: +X is to the front, +Y is to the left, and +Z is up.



When the named audio stream is given a 3D position, the service filters add the audio to its 3D soundfield, which is then listened to using an AudioIO/Headphone3DOut or a set of AudioIO/ SpeakerOut components.

Speaker Arrays

Add AudioIO/SpeakerOut components for each speaker in the speaker array, specifying the position of each speaker relative to a reference point. Currently, a highway must be set in each SpeakerOut. The highway must be one of the first 16 highways and shouldn't be used by any other SpeakerOut or HighwayOut component. In general, the more speakers, the higher the accuracy of the reconstructed soundfield. The service sends the appropriate audio to each SpeakerOut component, which is then linked to a IO interface component such as an ACU or AMP channel.

Headphones

Add a Headphone3DOut component for operators that will have 3D audio in their headsets. If a head-tracker is available, the component accepts azimuth and elevation inputs that specify a rotation applied to the soundfield to match the head's orientation. The azimuth and elevation values should indicate the orientation in degrees relative to facing directly forward. For example, 0 azimuth and 0 elevation is straight forward, while 90 azimuth and 45 elevation is looking directly to the left then up 45 degrees. Link the two audio outputs (left and right) to the two Io interface components such as ACU or ACE-RIU channels used for the operator's stereo headphones.

4.10. Automated Speech Recognition

Automated speech recognition (SR) is a natural extension to the ASTi communications modeling environment and is easily added to ACE models. ACE signal processing components are available to pre-process speech to ensure the audio is at an ideal volume, unnecessary frequency content is removed, and a high signal-to-noise ratio. Any audio stream in a model can feed into the SR system, whether it's a microphone, a radio or intercom, or any other mix. The result is a tightly integrated communications and SR system with significant power and flexibility.

ACE SR is useful in a number of roles in simulators:

- Instructor workstation voice control
- Cockpit system voice control
- Automated air traffic control
- Radio environment simulation
- Talk-on-target training systems

Up to 16 simultaneous SR streams are supported on one T4 Target platform. A SpeechFeed component in the model acts as the interface between the communications system and a SR stream. Streams are typically allocated on a per operator or per radio or intercom basis.

Custom grammars are built to match the vocabulary and phraseology for each simulation. These grammars are part of an ACE project, like communications models. Several tools are available to assist with grammar development, including grammar compilers, phrase generation utilities, and testing tools. A key feature of grammars in ACE is that they extract the *meaning* of what was spoken, in addition to recognizing the words themselves.

An "SR plan" file acts as the central configuration point for speech recognition in an ACE layout. The SR plan specifies host network settings (where to send the recognition results), which grammars to use on each stream, and various parameters to tune the underlying SR engine.

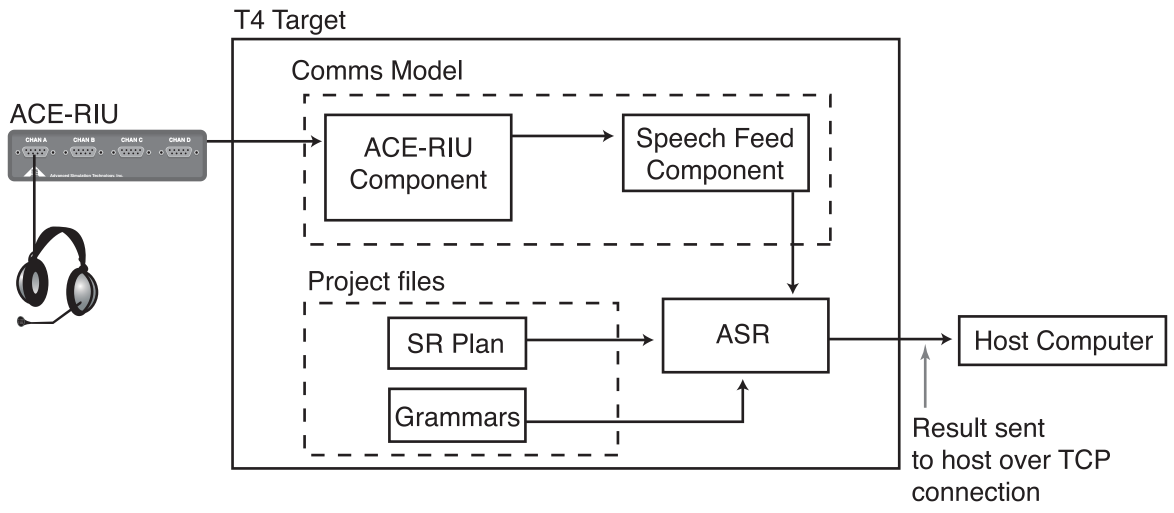


Figure 3: Automated Speech Recognition Architecture

Grammar Files

Grammar files are the specifications for the words and phrases that are recognized by the SR system. In ACE, a single SR stream can have one or more grammars active at a time. Manage the grammar files at runtime via the SR host interface, which supports operations like activating and deactivating grammars on a per stream basis.

Typically, there is one grammar for each type of role that requires SR in a simulator. For example, let's say there is a grammar file for an instructor station that uses SR for simulation commands, and another grammar file to recognize speech from a student pilot. If the student has multiple scenarios to practice (asking for takeoff clearance, designating a target, etc.), there could be a grammar for each scenario. Default grammars files for each SR stream are specified in the SR plan in the project.

There are several grammar formats and different representations are generally equivalent although some are more concise than others. The W3C standard for grammars is the Speech Recognition Grammar Specification (SRGS) which discusses two main approaches, BNF and XML. XML is more widely used than BNF and is the format of choice for use in the ACE development environment. Another W3C specification, the Semantic Interpretation for Speech Recognition (SISR), details the use of semantic tags in SRGS grammars in order to extract the meaning from speech. Information and techniques covered in these W3C specifications is generally applicable when using SR in ACE.

The W3C SRGS specification is available here:

```
http://www.w3.org/TR/speech-grammar/
```

The SISR recommendation is available here:

```
http://www.w3.org/TR/semantic-interpretation/#SI8
```

When a grammar file is first created in ACE, a simple template is added that matches the phonetic alphabet (alpha, bravo, etc.) and digits (0-9). The template demonstrates basic usage of commonly used features from the W3C specifications and is a good starting point for learning to build grammars.

Several useful built-in grammars are also available that can be embedded in new, custom grammars. The built-in grammars include:

- Alphanumeric strings
- Dates
- Digits (“one two three four”)
- Numbers (“one thousand two hundred and thirty four”)
- Time

Recognition Phases

The recognition engine is at the core of ACE SR. The engine receives audio from the model, analyzes it, matches speech to the active grammar, and returns its best guesses about what was spoken. The engine features:

- Speaker independent recognition (no training required)
- Continuous recognition (pausing between words is unnecessary)
- Background noise robustness
- Tunable parameters for timing, levels, etc.
- Dynamic grammar switching

To properly configure SR for a given application, some knowledge of the SR engine's parameters and operation is needed. The default SR plan in the project exposes the most common tuning parameters, including the confidence threshold, NBest list length, and EOS and BOS backoffs. Each guess the engine makes has a confidence value (0-100%). If the confidence level is below the confidence threshold, the guess is not sent to the host. Usually this is left at 0%, to allow the host to decide what to do with a recognition, even if it was a low confidence. The NBest list length sets the number of guesses that are sent to the host for each recognition attempt. Usually this value is left at 1 to get only the best guess.

The EOS (end-of-speech) and BOS (beginning-of-speech) backoffs define safety margins (in milliseconds) to ensure all of the speech is captured when separating speech from the surrounding audio (noise or silence). EOS backoff and incomplete timeout together describe how long to wait before speech is declared “finished”, and consequently how quickly a recognition result becomes available. Longer EOS backoffs and incomplete timeouts allow for longer pauses between words, while short backoffs and timeouts deliver quick responses. Set too short, these parameters will cause phrases to be chopped up into multiple recognitions or cause recognition to fail. An EOS backoff in the range of 350 to 1000 ms is a good starting point for most applications. There is no one-size-fits-all group of settings. Tune the parameters based on the duration of speech being recognized (words vs. phrases vs. paragraphs), the simulation environment (noisy or not), and so on.

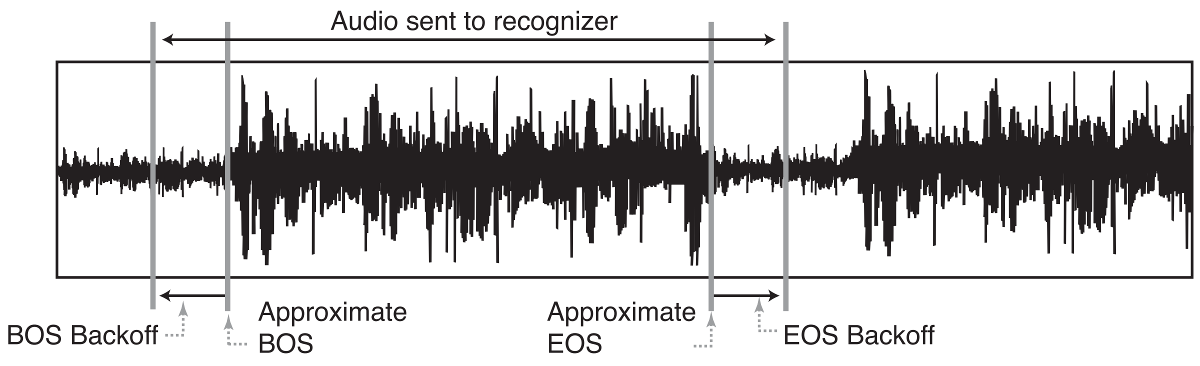


Figure 4: Speech Recognition Timing

Examples of other available parameters:

- Control noise adaptation
- Set sensitivity of the engine to sounds (clicks, pops, and tones)
- Control SNR requirements
- Log waveforms of recognized phrases

Contact ASTi for more information on tuning the SR engine.

Host Interface

Speech recognition in ACE recognizes what is spoken and can extract the meaning of the speech, but deciding what to do about it is an entirely different matter. In some SR use cases, like IOS control, the decision making is straightforward. The meaning of the speech might directly map to commands or functions in the instructor control station software. For example, saying “freeze” might freeze the simulation. However, in situations like automated role-playing, there is much greater complexity, possibly involving rule engines or artificial intelligence.

The ACE SR host interface sends TCP packets to the host computer containing information the host needs to implement the decision-making logic. The packet ICD is described below. It includes the recognition text (the “literal”), the meaning, the confidence, and so on. Packets are sent when speech is recognized, and, optionally, when speech starts and stops. Not all fields in the ICD are filled in (since they aren't applicable) for the speech start and stop messages. The host should open a TCP server socket and wait to accept a connection from the SR system on the Target.

The SR plan “[HOST]” section lists the possible configuration parameters for the host interface. This includes the host IP address and port (the host should open a TCP server socket and wait for a connection on this port). The “device” is the ethernet interface on the Target. Obviously, pick the one that is on the same network as the host computer. The “events” option enables or disables event messages for speech start and stop. Continue with this section to view an example SR plan.

Host Interface ICD

stream_id - The SR stream this packet is about.

type - The type of packet this is:

1 = Recognition succeeded

2 = Recognition failed

3 = Speech started

4 = Speech finished

5 = Recognition error

result_num - The amount of recognitions that have happened so far.

num_guesses - The number of guesses to send for this recognition.

confidence - A 0-100 confidence rating for this guess.

rec_status - Status of the engine when recognition completes.

bos - Sample index where the beginning of speech was found. -1 if the BOS has not yet been found.

eos - Sample index where the end of speech was found. -1 if EOS has not yet been found. This will only work if swiep_mode is set to begin_end.

timestamp - Clock time on the Target when this event happened, in seconds since epoch.

grammar - The grammar used for this recognition.

literal - The recognized phrase.

meaning - The meaning string for the phrase, as built in the grammar.

Host Interface Packet Details

Variable	Offset	Type	Size (in bytes)
stream id	0	unsigned int	4
type	4	unsigned int	4
result num	8	unsigned int	4
num guesses	12	unsigned int	4
confidence	16	unsigned int	4
rec status	20	unsigned int	4
bos	24	int	4
eos	28	int	4
timestamp	32	double	8
reserved	40	double	8
reserved	48	double	8
grammar	56	char[]	32
literal	88	char[]	512
meaning	600	char[]	512
reserved	1112	char[]	128
		Total Size:	**1240 Bytes**

Host Command Interface

An SR command interface is also available that provides remote control of SR streams and grammars. The main uses are enabling or disabling a stream and switching grammars to support a different recognition scenario. The interface uses the same host socket connection as previously discussed.

Host commands are strings, terminated with a new line character, that have a command line options-like syntax quite similar to the TTS interface. Multiple options can be specified in one command, and, unlike standard command line options, the order does matter. A command commonly contains a sequence of actions, such as, "On stream 1: disable the stream, deactivate grammar X, activate grammar Y, enable the stream". By disabling the stream, switching grammars, and enabling the stream, the command guarantees that the grammars become active immediately. If the stream is not disabled prior to grammar switching commands, the grammars won't switch until the current recognition finishes.

Command Options

Variable	Offset
-s STREAM	The stream number this command applies to
-e	Enable the stream
-d FLAG	Disable the stream. If FLAG is “1”, any recognition result generated when the stream is disabled will be thrown away (not sent to the host). If FLAG is “0”, any result will not be thrown away.
-l GRAMMAR	Load a grammar into memory. This is done implicitly when a grammar is activated, so the load and unload commands are usually unnecessary.
-u GRAMMAR	Unload a grammar
-a GRAMMAR	Activate a grammar
-v GRAMMAR	Deactivate a grammar
-w	Deactivate all grammars

Example Commands

Command	“-s l -d l -w -a ios_grammar -e”
Meaning	Disable stream 1 immediately, deactivate all its grammars, activate ios_grammar, and then enable the stream. Ignore any recognition results generated when disabling the stream.

Command	“-s 3 -v scenario1 -s scenario2”
Meaning	When the current recognition on stream 3 completes, deactivate the scenario1 grammar and activate the scenario2 grammar.

Miscellaneous Tips

Use the Scope to view the signal in the SpeechFeed component. Speech audio should clearly be visible, but not going beyond -1.0 to 1.0 (which would cause clipping). Consider using the Dynamics components, such as AGC, to achieve consistent volume levels even when the microphone is close or far away.

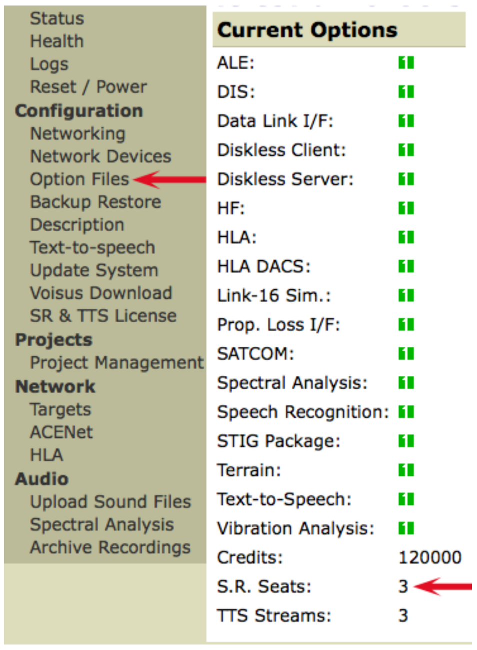
Use one or more Audio/Filter or EnvCue/Multifilter components to eliminate unneeded frequencies in the speech signal. A highpass at 50 Hz and a lowpass at 4 KHz are typical.

Use the sr_host.py host script for debugging when the real host is not up and running.

Experiment with grammars to find what works best. Alternative pronunciations (e.g. “want to” vs. “wanna”) and optional filler words may boost confidence levels. Limit the allowed phraseology as much as possible in the grammar - the fewer options, the higher the confidence. Build user pronunciation dictionaries to accommodate alternative pronunciations.

SR Operational Steps

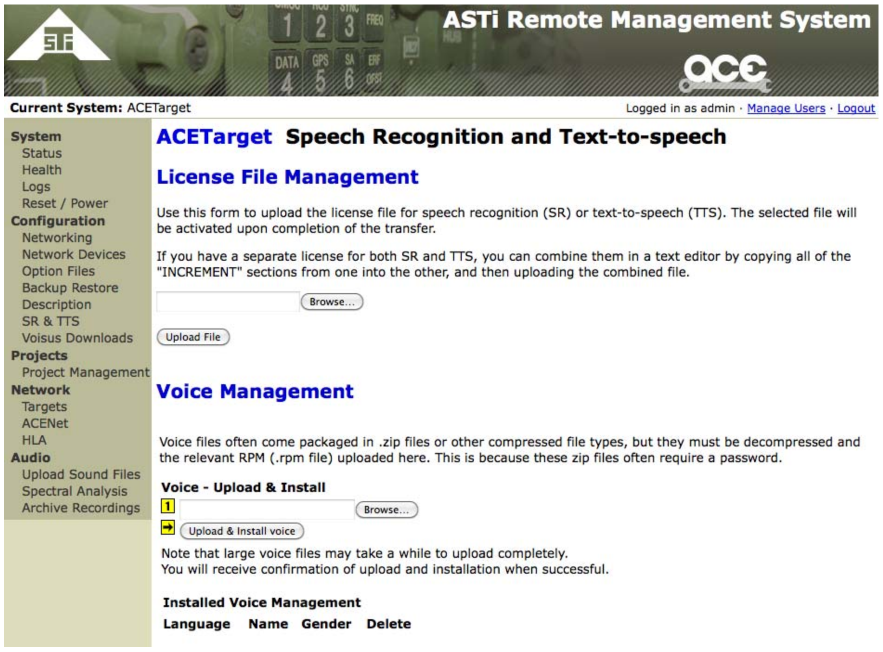
1. In RMS, upload an option file that enables Speech Recognition.



Current Options

ALE:	🟩
DIS:	🟩
Data Link I/F:	🟩
Diskless Client:	🟩
Diskless Server:	🟩
HF:	🟩
HLA:	🟩
HLA DACS:	🟩
Link-16 Sim.:	🟩
Prop. Loss I/F:	🟩
SATCOM:	🟩
Spectral Analysis:	🟩
Speech Recognition:	🟩
STIG Package:	🟩
Terrain:	🟩
Text-to-Speech:	🟩
Vibration Analysis:	🟩
Credits:	120000
S.R. Seats:	3
TTS Streams:	3

2. In RMS, upload a license file via the SR & TTS page.



Current System: ACETarget

Logged in as admin · Manage Users · Logout

ACETarget Speech Recognition and Text-to-speech

License File Management

Use this form to upload the license file for speech recognition (SR) or text-to-speech (TTS). The selected file will be activated upon completion of the transfer.

If you have a separate license for both SR and TTS, you can combine them in a text editor by copying all of the "INCREMENT" sections from one into the other, and then uploading the combined file.

Voice Management

Voice files often come packaged in .zip files or other compressed file types, but they must be decompressed and the relevant RPM (.rpm file) uploaded here. This is because these zip files often require a password.

Voice - Upload & Install

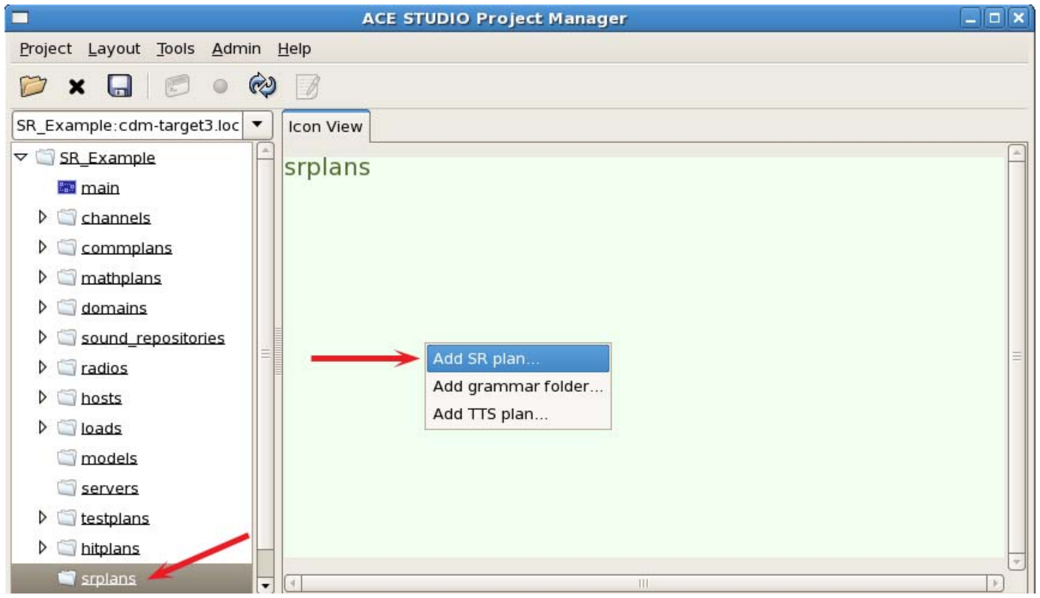
Note that large voice files may take a while to upload completely.
You will receive confirmation of upload and installation when successful.

Installed Voice Management

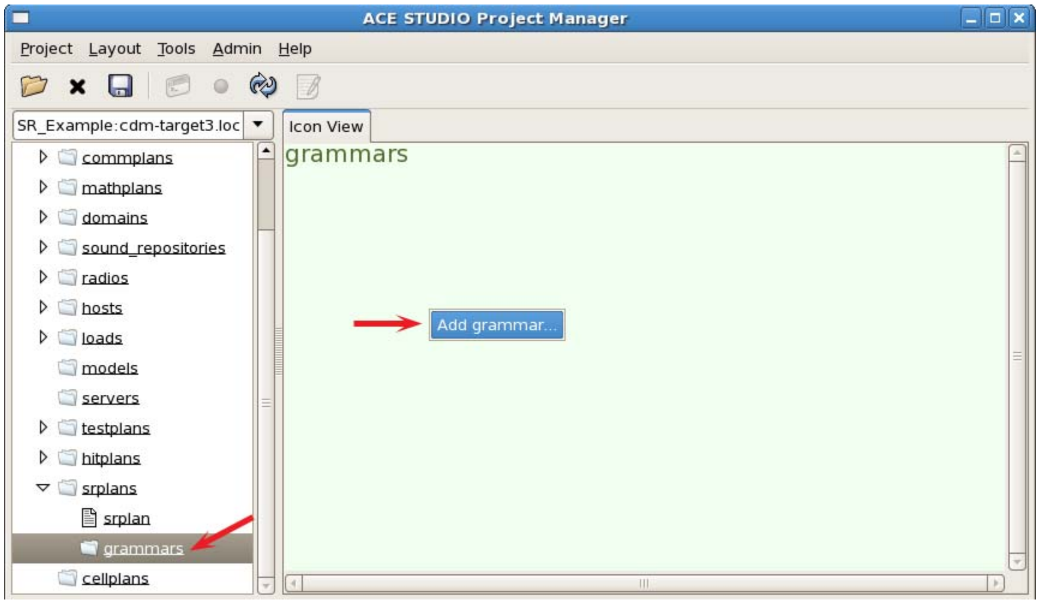
Language	Name	Gender	Delete

3. Reboot the Target.
4. Open a project in ACE Studio.

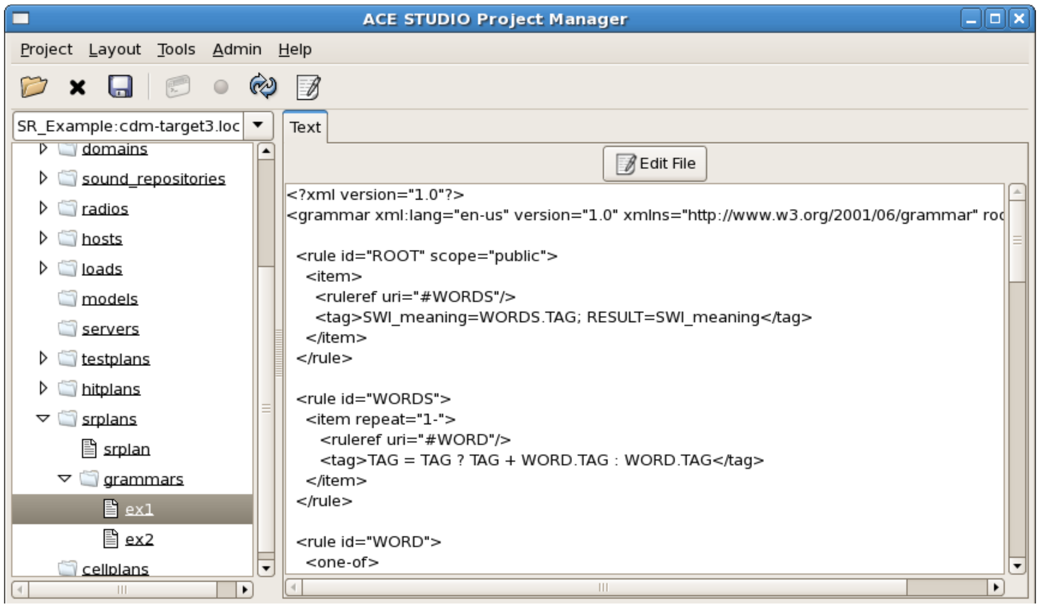
5. Click on the ‘srplans’ folder in the tree view, then right click on the canvas to add an SR plan and a grammar folder.



6. Click on the grammar folder in the tree view, then right click on the canvas to add individual grammar files.

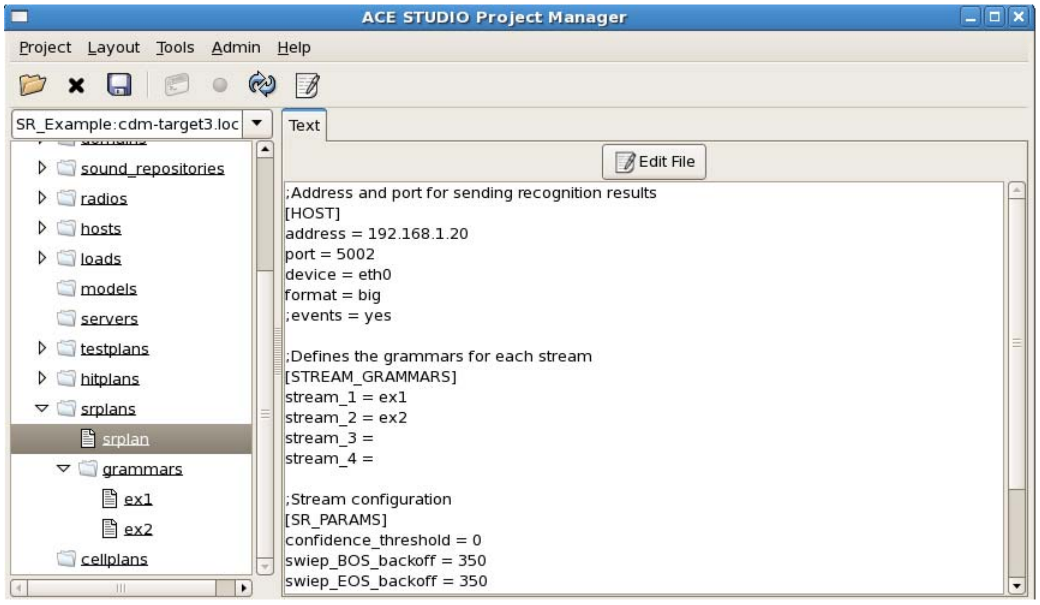


7. Edit the grammar files to define the words and structure of speech that will be recognized.



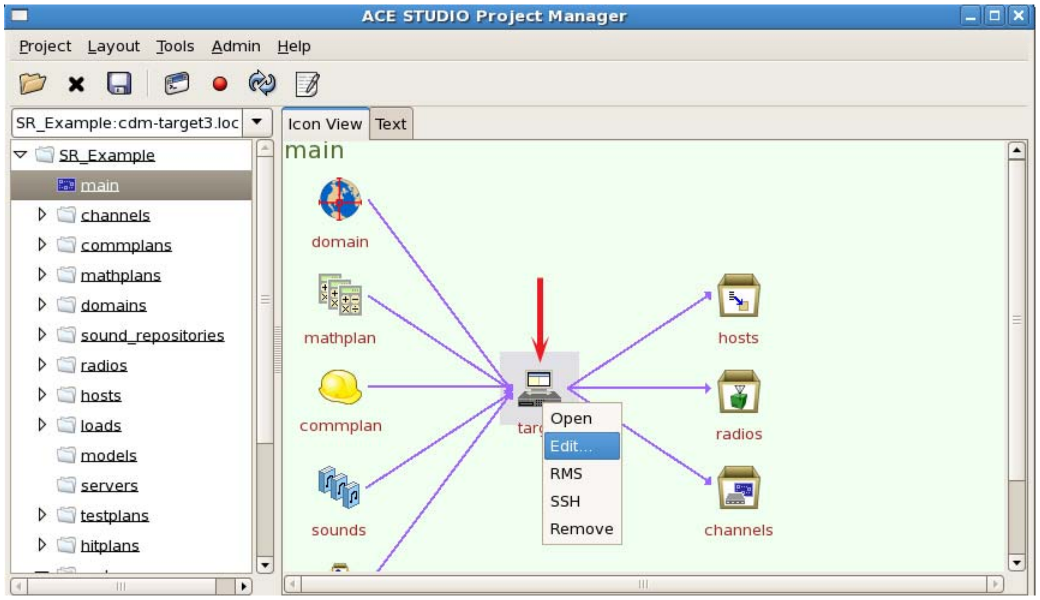
```
<?xml version="1.0"?>
<grammar xml:lang="en-us" version="1.0" xmlns="http://www.w3.org/2001/06/grammar" roo

 <rule id="ROOT" scope="public">
   <item>
     <ruleref uri="#WORDS"/>
     <tag>SWI_meaning=WORDS.TAG; RESULT=SWI_meaning</tag>
   </item>
 </rule>

 <rule id="WORDS">
   <item repeat="1-">
     <ruleref uri="#WORD"/>
     <tag>TAG = TAG ? TAG + WORD.TAG : WORD.TAG</tag>
   </item>
 </rule>

 <rule id="WORD">
   <one-of>
```

8. Edit the SR plan to specify host connection settings, the default grammar for each stream, and various other parameters. Use the “Edit File” button to open the standard text editor. Clicking ‘save’ in the editor window will save to the file in the project. See the screen below for an example of how to specify grammars.

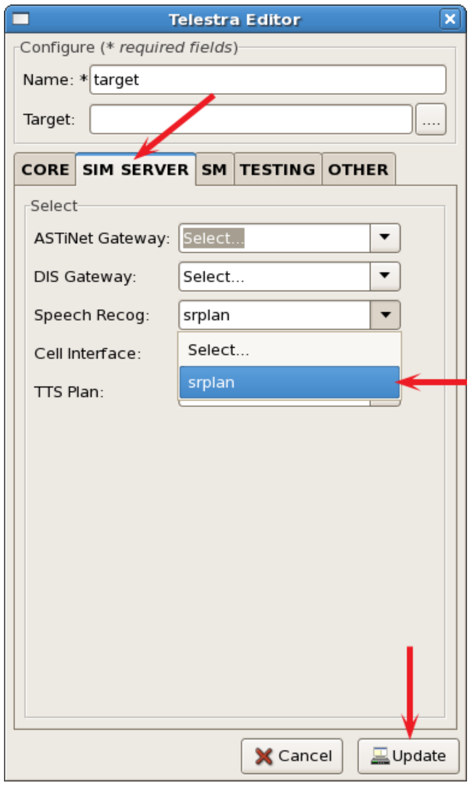


```
;Address and port for sending recognition results
[HOST]
address = 192.168.1.20
port = 5002
device = eth0
format = big
;events = yes

;Defines the grammars for each stream
[STREAM_GRAMMARS]
stream_1 = ex1
stream_2 = ex2
stream_3 =
stream_4 =

;Stream configuration
[SR_PARAMS]
confidence_threshold = 0
swiep_BOS_backoff = 350
swiep_EOS_backoff = 350
```

9. In the layout, open the Telestra Editor window by right-clicking on the Target icon and selecting "Edit…".



10. Select the SIM SERVER tab. In the Speech Recog drop-down menu select the srplan. Click "Update" to close the Telestra Editor window.



Name: * target

Target:

Select

ASTiNet Gateway: Select...

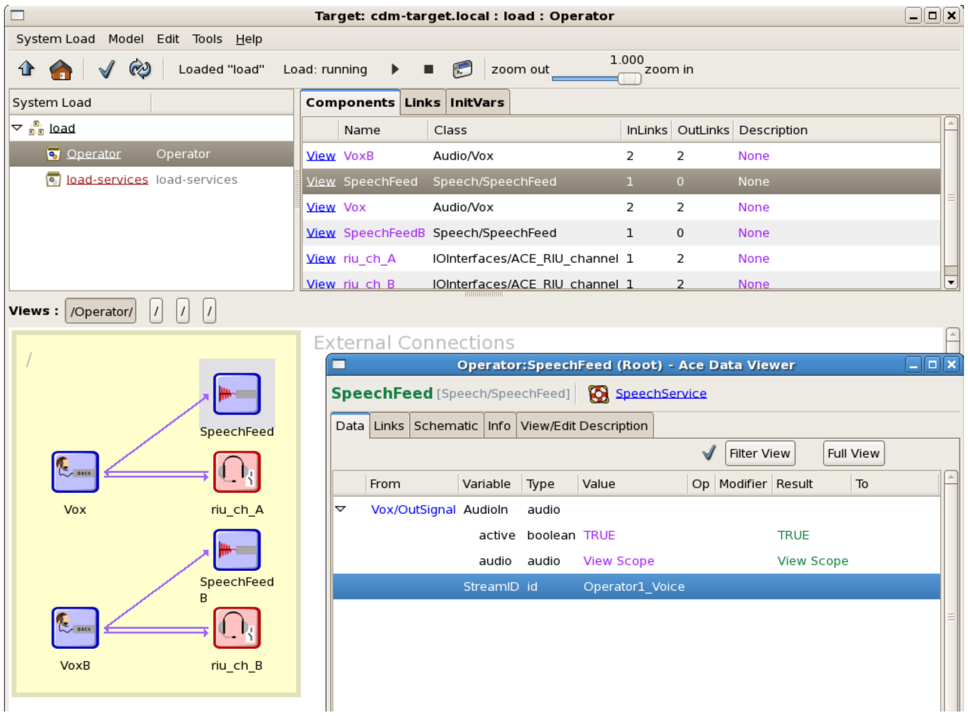
DIS Gateway: Select...

Speech Recog: srplan

Cell Interface:

TTS Plan:

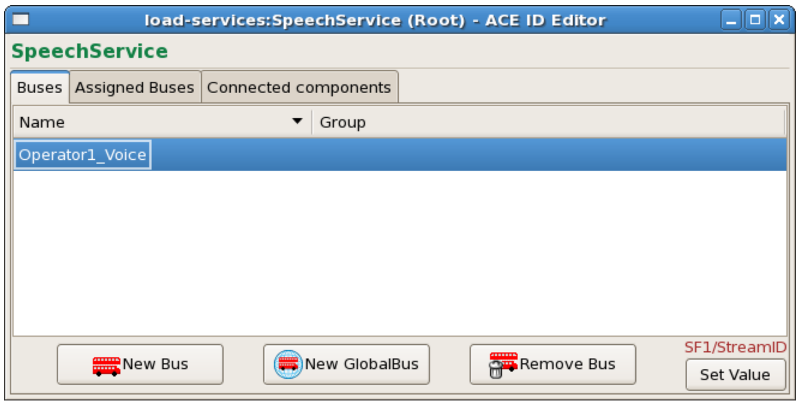
11. Install the layout, open the load viewer, and create a new sim model. Link the audio input (microphone) from an ACENet channel component to the Speech > SpeechFeed component. Optionally, add an Audio > Vox component to only allow audio through when a PTT switch is pressed.


Target: cdm-target.local : load : Operator
System Load Model Edit Tools Help
Loaded "load" Load: running zoom out 1.000 zoom in
System Load
load
Operator Operator
load-services load-services
Components Links InitVars
Views : /Operator/ / / /
External Connections
Vox VoxB SpeechFeed SpeechFeed B riu_ch_A riu_ch_B
Operator:SpeechFeed (Root) - Ace Data Viewer
SpeechFeed [Speech/SpeechFeed] SpeechService
Data Links Schematic Info View/Edit Description
Filter View Full View

	Name	Class	InLinks	OutLinks	Description
View	VoxB	Audio/Vox	2	2	None
View	SpeechFeed	Speech/SpeechFeed	1	0	None
View	Vox	Audio/Vox	2	2	None
View	SpeechFeedB	Speech/SpeechFeed	1	0	None
View	riu_ch_A	IOInterfaces/ACE_RIU_channel	1	2	None
View	riu_ch_B	IOInterfaces/ACE_RIU_channel	1	2	None

From	Variable	Type	Value	Op	Modifier	Result	To
Vox/OutSignal	AudioIn	audio					
	active	boolean	TRUE			TRUE	
	audio	audio	View Scope			View Scope	
	StreamID	id	Operator1_Voice				

12. Open the SpeechFeed component, double-click “UNASSIGNED”, add a new SpeechService bus called “Operator1_Voice”, and click “Set Value”.

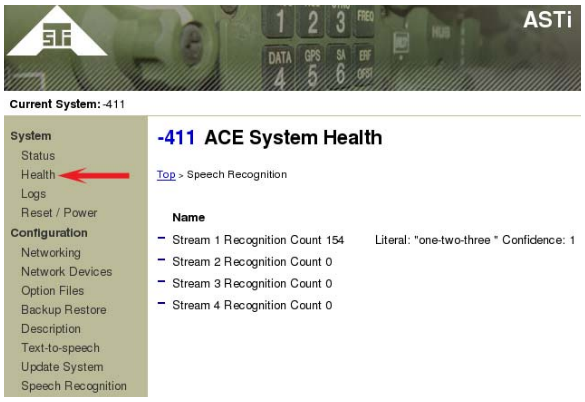

load-services:SpeechService (Root) - ACE ID Editor
SpeechService
Buses Assigned Buses Connected components
New Bus New GlobalBus Remove Bus
SF1/StreamID
Set Value

Name	Group
Operator1_Voice	

13. Apply changes and reinstall the layout.

14. To receive speech recognition results, open a terminal on either the Studio or Target, and enter the command “sr_host.py”. This script accepts a connection from the SR daemon running on the Target and prints any packets it receives. The SR plan host address must be the address of the computer running the host script.

15. Some SR status and debug information is available in the RMS Health system. Click on the Health page, then on the "Speech Recognition" and "Speech Streams" links to view more information.



-411 ACE System Health

Top > Speech Recognition

Name
- Stream 1 Recognition Count 154 Literal: "one-two-three " Confidence: 1
- Stream 2 Recognition Count 0
- Stream 3 Recognition Count 0
- Stream 4 Recognition Count 0

Example SR Plan

```
; Comment lines begin with a semicolon
;
[HOST]
; Address and port for sending recognition results
address = 192.168.1.20
port = 5002
device = eth0
format = big
; Do not send event notifications for speech started, stopped
events = no

;Defines the default grammars for each stream
[STREAM_GRAMMARS]
stream_1 = ex1
stream_2 = ex2
stream_3 =
stream_4 =

;Stream configuration
[SR_PARAMS]
; Send recognitions to the host, no matter how low the confidence
confidence_threshold = 0
swiep_BOS_backoff = 350
; Use 350 ms as the end-of-speech timeout
swiep_EOS_backoff = 350
swiep_mode = begin_only
; Return only the best guess
swirec_nbest_list_length = 1
```

Simple GRXML Grammar File Example

The following grammar recognizes any combination of the words “one”, “two”, “three”, “alpha”, “bravo”, and “charlie”. Recognition begins at the “ROOT” rule. Rulerefs are used to refer to other rules within a rule. ROOT refers to the WORDS rule in this way. WORDS uses <item repeat=“1-”> to match one or more single words. The WORD rule uses <one-of> to match either digits or callsigns.

Tags are used throughout grammar files to associate a meaning with the recognized words or phrases. Tags are actually snippets of javascript code, which gives great flexibility when building up a meaning or understanding what was spoken. The tag strings can be passed between rules, allowing the concatenation of meanings when a phrase is spoken. In this example, numbers are mapped to a digit meaning (“one” has a meaning of “1”) and callsigns are mapped to a letter meaning (“alpha” has a meaning of “A”). Consequently, the phrase “one two alpha two” will have a returned meaning of “12A2”. Generally, it's recommended that the host code makes decisions based on the meaning string, rather than the literal string. Meaning strings can be constructed in an easy-to-parse format and can be used to extract only relevant information from speech. Contact ASTi for more details.

Using a combination of the above techniques, grammars can be built to recognize a variety of complex, structured speech patterns.

```
<?xml version=“1.0”?>
<grammar xml:lang=“en-us” version=“1.0” xmlns=”http://www.w3.org/2001/06/grammar”
root=“ROOT">
 <rule id=“ROOT” scope=“public”>
  <item>
   <ruleref uri=“#WORDS”/>
   <tag>SWI_meaning=WORDS.TAG; RESULT=SWI_meaning</tag>
  </item>
 </rule>
 <rule id=“WORDS”>
  <item repeat=“1-”>
    <ruleref uri=“#WORD”/>
    <tag>TAG = TAG ? TAG + WORD.TAG : WORD.TAG</tag>
  </item>
 </rule>
 <rule id=“WORD”>
  <one-of>
   <item>
```

```
      <ruleref uri="#DIGIT"/>
      <tag>TAG=DIGIT.TAG</tag>
    </item>
    <item>
      <ruleref uri="#CALLSIGN"/>
      <tag>TAG=CALLSIGN.TAG</tag>
    </item>
  </one-of>
</rule>
  <rule id="DIGIT">
  <one-of>
    <item> one   <tag>TAG='1'</tag></item>
    <item> two   <tag>TAG='2'</tag></item>
    <item> three <tag>TAG='3'</tag></item>
  </one-of>
</rule>
<rule id="CALLSIGN">
  <one-of>
    <item> alpha    <tag>TAG='A'</tag></item>
    <item> bravo    <tag>TAG='B'</tag></item>
    <item> charlie  <tag>TAG='C'</tag></item>
  </one-of>
</rule>
</grammar>
```

4.11. Text-To-Speech Voice Messaging Systems

Introduction

Simulators, and training systems in general, use voice messaging in two broad categories; the first is the replication of real vehicle systems, while the second is the generation of voice signals heard via radio or (less frequently) intercom reception.

There are two approaches used to generate the required voice messages. Conventionally, sound files have been used either playing complete messages, or using sound files for individual words or phrases that are stitched together to form the required message. Alternatively, a text to speech (TTS) generator may be used to create the message.

Vehicle Systems

To replicate a vehicle voice message system, it is generally necessary to use recordings of the original system and replay them as sound files. It is usually important that the character, timing, tone, and fidelity of the original system be maintained so that the crew hears the signals as they would in the vehicle. There may be exceptions to this, particularly where the function of the simulator is more mission-oriented and less a trainer directed at the operation of the vehicle itself.

Typical vehicle systems include the following:

- TCAS – Traffic Collision Avoidance System
- EGPWS/GPWS – [Enhanced] Ground Proximity Warning System
- PWS – Predictive Windshear System
- VMU – Voice Message Unit (F-16)
- RWR – Radar Warning Receiver (various)

Radio Messaging Systems

In addition to normal radio communications, a number of radio messaging systems exist in the real world that may be required to support realism within a simulated environment. This class of system is transmitted over the same radios as used for normal voice communications.

ATIS

The most common requirement for simulators is provision of ATIS, or Automatic Terminal Information System messages. These messages provide non-control information in terminal areas (i.e. airports), and comprise weather, runway, approach and optionally NOTAM (Notice To Airman) elements. These messages follow a predefined format, though there are regional variations.

A typical message would have the following structure:

> this is <AIRPORT NAME> <ARRIVAL / DEPARTURE> information <DESIGNATOR>
>
> main takeoff runway <RUNWAY NUMBER>
>
> main landing runway <RUNWAY NUMBER>
>
> wind <WIND DIRECTION IN DEGREES> degrees <WIND FORCE IN KNOTS> knots
>
> visibility <VISIBILITY IN KILOMETERS> kilometers
>
> view <VIEW IN FEET> feet
>
> temperature <AIRPORT TEMPERATURE IN DEGREES CELSIUS> dew point <DEW POINT TEMPERATURE IN DEGREES CELSIUS>
>
> qnh <QNH PRESSURE IN HECTO PASCAL> hector pascal
>
> end of information <DESIGNATOR>

In the real world many airports are switching over to automated ATIS systems using text-to-speech to generate the messages. Alternatively, simply recording a person reading the text produces the message. This is problematical since the message must be updated anytime the weather or other reported parameter changes, and may occur several times in the period of one hour.

VOLMET

This is the name given to a series of HF radio stations dedicated to providing TAF (Terminal Aerodrome Forecast), SIGMET (significant metrological information) and METAR (aviation routine weather report). The radio stations are positioned worldwide to provide international coverage, and report weather for 20+ locations (airports, en-route waypoints, etc) on a periodic basis (typically every 30 minutes) in a cyclic order.

This is a less commonly utilized service, and is therefore not routinely required for simulators. Where it is defined, the use of TTS would be appropriate.

GCA – Ground Controlled Approach

This refers to a service provided by ground controllers at an airfield providing approach guidance to aircraft in marginal weather conditions. While ILS and GPS have largely obsoleted this technique in normal day-to-day operations, there are situations such as during equipment malfunctions where the use of GCA is the only option.

Commands are issued to the aircraft at least every 5 seconds, but often more frequently, and will take the following format:

> "well right of the on course correcting rapidly - fly heading 285"
> "9 miles from touchdown - fly heading 270"
> "intercepting the on course, fly heading 260"
> "on course - 8 miles from touchdown"
> "left of the on course - correcting slowly" (or "nicely", or "rapidly")
> "drifting left of course, turn right heading 262"
> "well left of the oncourse - turn right heading 265"
> "on course fly heading 262 - 7 miles from touchdown - 2 miles to glidepath interception"
> "slightly left of the on course - standby for glidepath interception - wheels should be down"
> "intercepting glidepath - commence descent for a 3° glidepath NOW - published decision height 327 feet"
> "initial rate of descent has you slightly below glidepath, adjust rate of descent"
> "on course"
> "above glidepath - adjust rate of descent"
> "left of the on course, fly heading 264 - 4 miles from touch down"
> "back on glidepath - resume normal rate of descent - on course"
> "below glidepath adjust rate of descent - 3 and one half miles from touch down - on course"
> "3 miles from touchdown - dangerously below the glidepath - level off your aircraft - acknowledge"

As will be apparent, the list of words/phrases is extensive and implementing this sub-system using sound files is complex and time consuming. Essentially this is very similar to the implementation of an ATIS system. The use of TTS would be applicable to implement this system.

4.12. Text-to-Speech in ACE

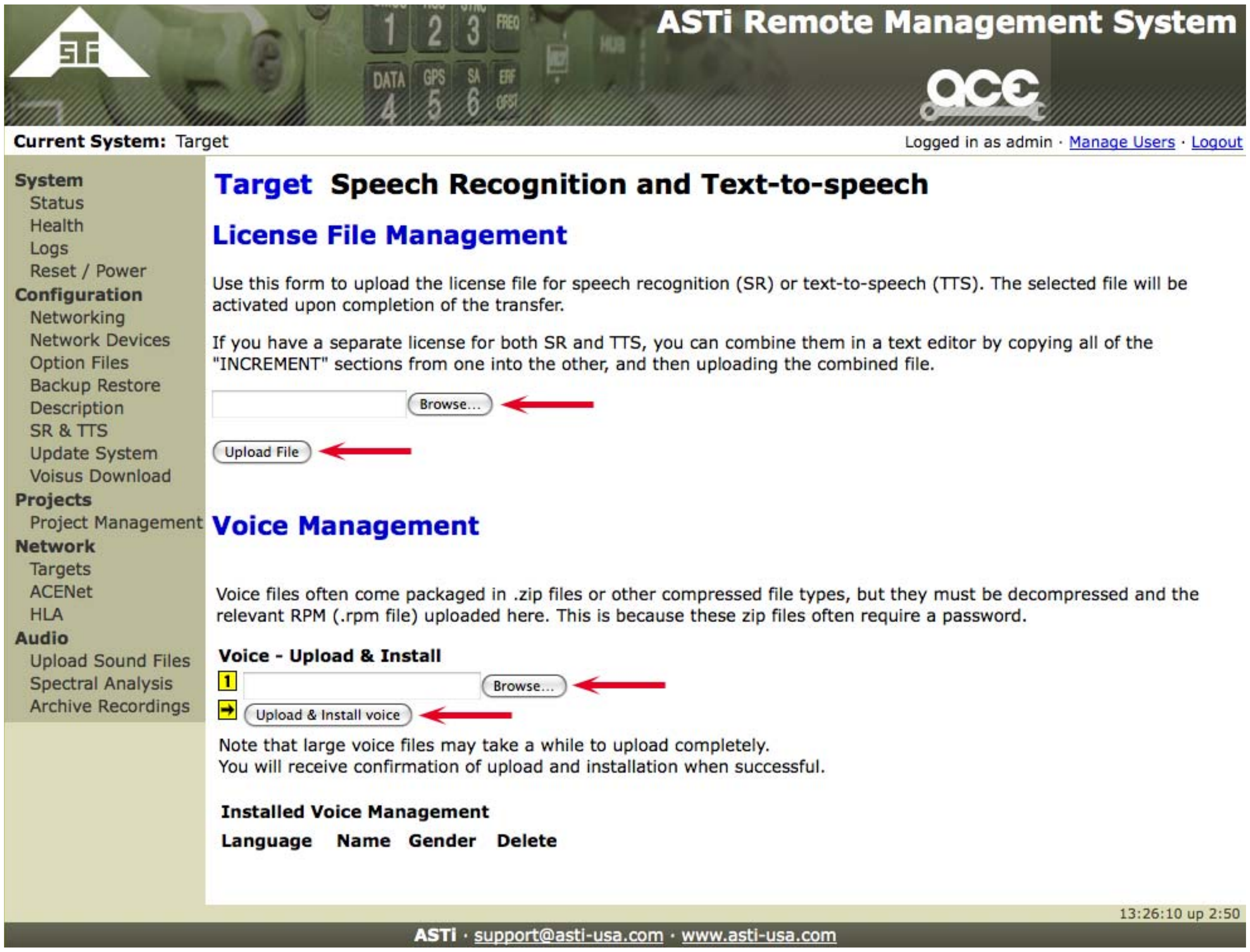
ACE Text-to-Speech (TTS) provides a simple interface for converting text into a speech audio stream in any ACE model. The model can then route the audio to any headset, speaker, or simulated radio.

Requirements

The Text-to-Speech software requires TTS enabled in the options file and an additional license file which allows a limited number of voice streams. The options file and license file are tied to the Target’s Eth0 MAC address. Individual voice RPM packages must also be installed through RMS. Voice RPMs may be several hundred megabytes in size and may be provided on a separate DVD.

Enabling TTS

1. In RMS, upload the license file from the SR & TTS page under “License File Management.” Contact ASTi to get a license file for Text-to-Speech.
2. In RMS, upload the voice RPMs from the SR & TTS page under “Voice Management.” Browse to the RPM file then upload and install each voice. Each voice may take several minutes to upload and install.
3. Reboot the Target.



Target Speech Recognition and Text-to-speech

License File Management

Use this form to upload the license file for speech recognition (SR) or text-to-speech (TTS). The selected file will be activated upon completion of the transfer.

If you have a separate license for both SR and TTS, you can combine them in a text editor by copying all of the "INCREMENT" sections from one into the other, and then uploading the combined file.

Voice Management

Voice files often come packaged in .zip files or other compressed file types, but they must be decompressed and the relevant RPM (.rpm file) uploaded here. This is because these zip files often require a password.

Voice - Upload & Install

Note that large voice files may take a while to upload completely.
You will receive confirmation of upload and installation when successful.

Installed Voice Management

Language	Name	Gender	Delete

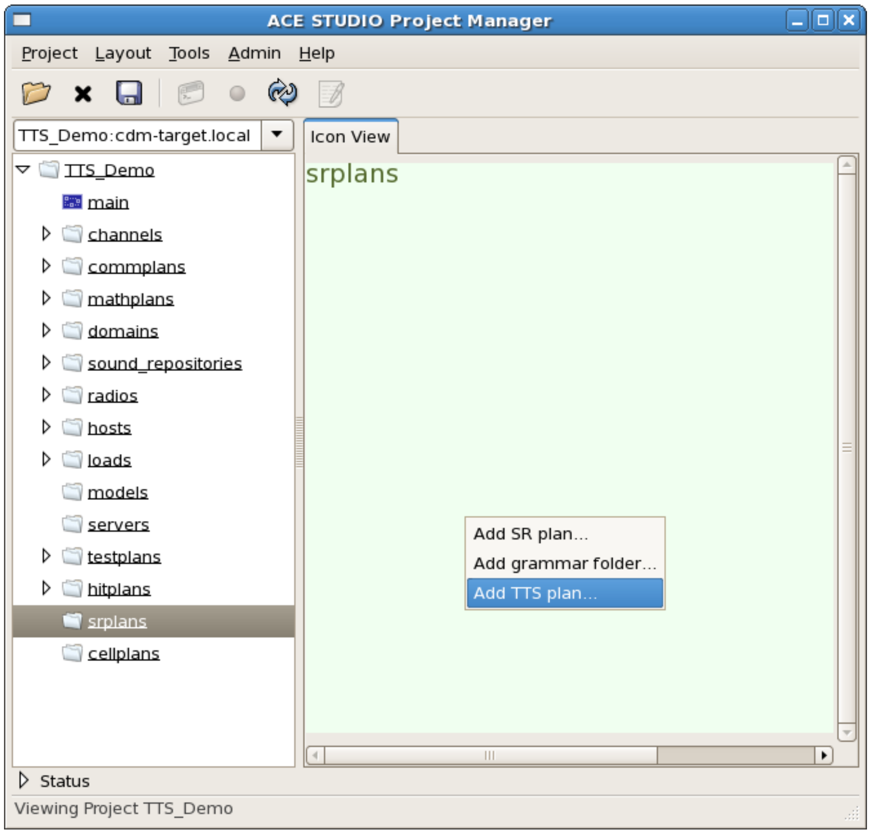
Host Interface Networking

The TTS system provides a TCP socket interface to the host computer. The host should create a TCP client socket and connect to the Target's IP address and TTS port. After connecting, the host can send commands to generate speech and receive notifications when speech starts or stops, or if there is an error. It is recommended that the host retain the TCP connection for the duration of the simulation rather than reconnecting for every command.

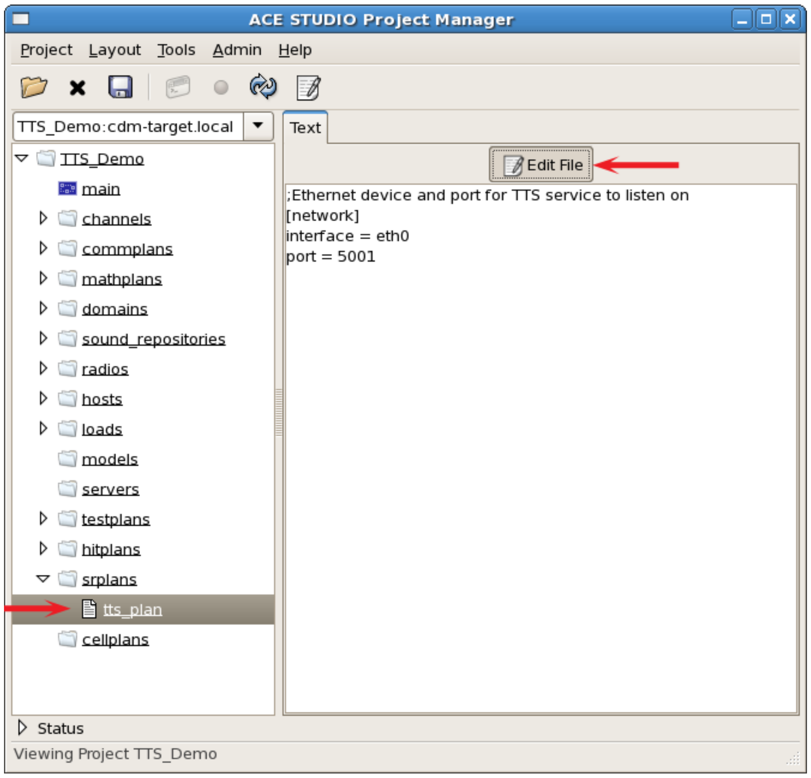
Setup

Follow these configuration steps to get TTS up and running.

1. In RMS, install an options file and license file with TTS enabled as well as at least one voice RPM (see previous page).
2. Add a TTS plan to a project by selecting the "srplans" folder and right-clicking in the icon view and selecting "Add TTS plan."

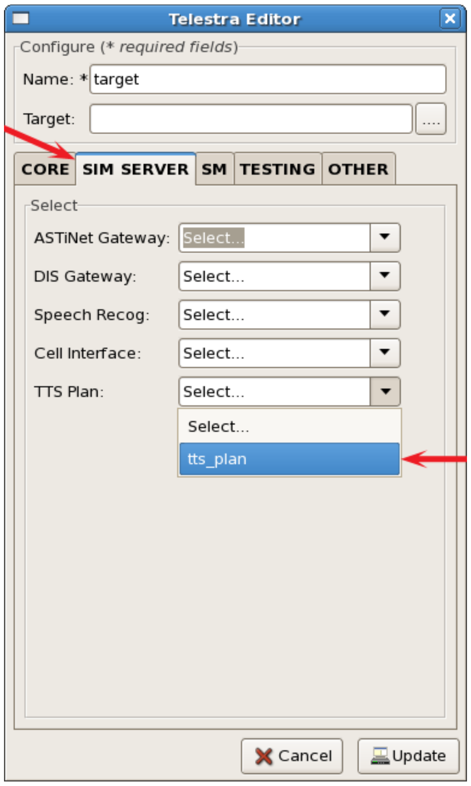


3. Modify the TTS plan to use the proper ethernet interface (default is eth0) and port (default is 5001).



```
;Ethernet device and port for TTS service to listen on
[network]
interface = eth0
port = 5001
```

4. Open the Telestra Editor and add the TTS plan to the layout.



Telestra Editor

Configure (* *required fields*)

Name: * target

Target:

Select

ASTiNet Gateway: Select...

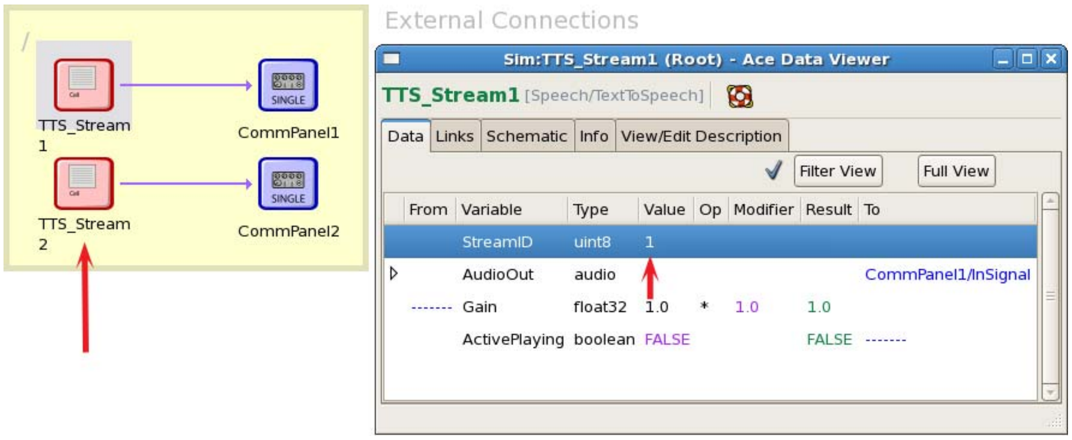
DIS Gateway: Select...

Speech Recog: Select...

Cell Interface: Select...

TTS Plan: Select...

5. Install the layout. Then add a Speech/Text-to-Speech component to a model and give it a stream ID of 1.



TTS_Stream1 [Speech/TextToSpeech]

From	Variable	Type	Value	Op	Modifier	Result	To
	StreamID	uint8	1				
	AudioOut	audio					CommPanel1/InSignal
-------	Gain	float32	1.0	*	1.0	1.0	
	ActivePlaying	boolean	FALSE			FALSE	-------

6. Send a command using the host interface. The Text-to-Speech component should output an active audio stream containing speech.

Speech/Text-to-Speech components can be added to any ACE model. Add one component for each TTS stream that is needed, and give each component a unique stream ID. Valid stream IDs start at 1 and go up to the number of streams available. It is an invalid configuration to have two Text-to-Speech components with the same stream ID. When a host command is received, speech will appear at the audio output of the corresponding Text-to-Speech component within a few hundred milliseconds. Link this audio output to any Mixer, Transceiver, CommPanel, or other component to route the speech into the simulation environment.

Customizing Speech

Two methods are available for customizing the speech generated by the TTS system: a markup format and user dictionaries. Using the markup format, tags are inserted into the speech text to change speech rate, volume, pronunciation, and other parameters. User dictionaries are simple text files on the Target's hard drive that specify words to be replaced with other words, or alternate pronunciations.

These features are documented in PDF files already installed on the Target's disk at:

```
/usr/local/ScanSoft/Realspeak_4.0/doc
```

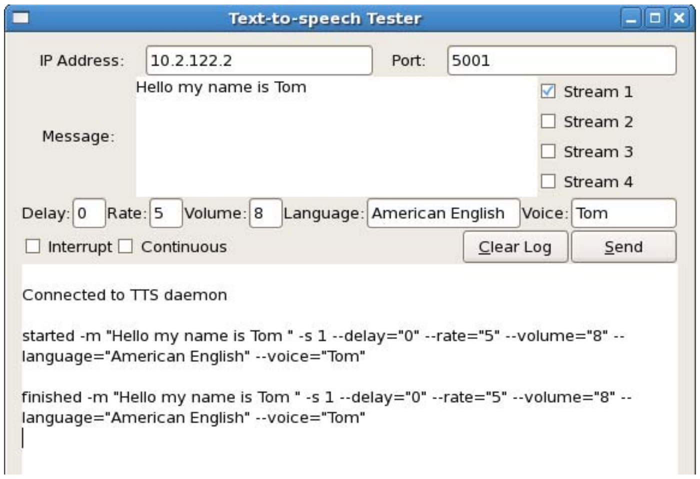
RS_User_Guide_ENU_V4.pdf - See Chapter 1 for information on using markup tags.

RS_Telecom_TTS_main_SDK_V40.pdf - See Chapter 3 for using user dictionaries.

TTS Tester

A TTS test application is installed on every ACE Studio platform. This application connects to the host interface and can send messages containing all the supported parameters like voice, rate, and language. Commonly this is used to test that the network interface and model are working in the absence of the simulation host computer. Type in a message, select the desired options, and click send. Responses from the TTS system are displayed in the text area at the bottom of the window.

To run the TTS Tester, open a terminal window and type "`tts_testgui.py`". Type "`tts_testgui.py -h`" to print the options available, which include shortcuts to set the IP address, port, and stream.



IP Address: 10.2.122.2 Port: 5001

Message: Hello my name is Tom

[✓ Stream 1]

Delay: 0 Rate: 5 Volume: 8 Language: American English Voice: Tom

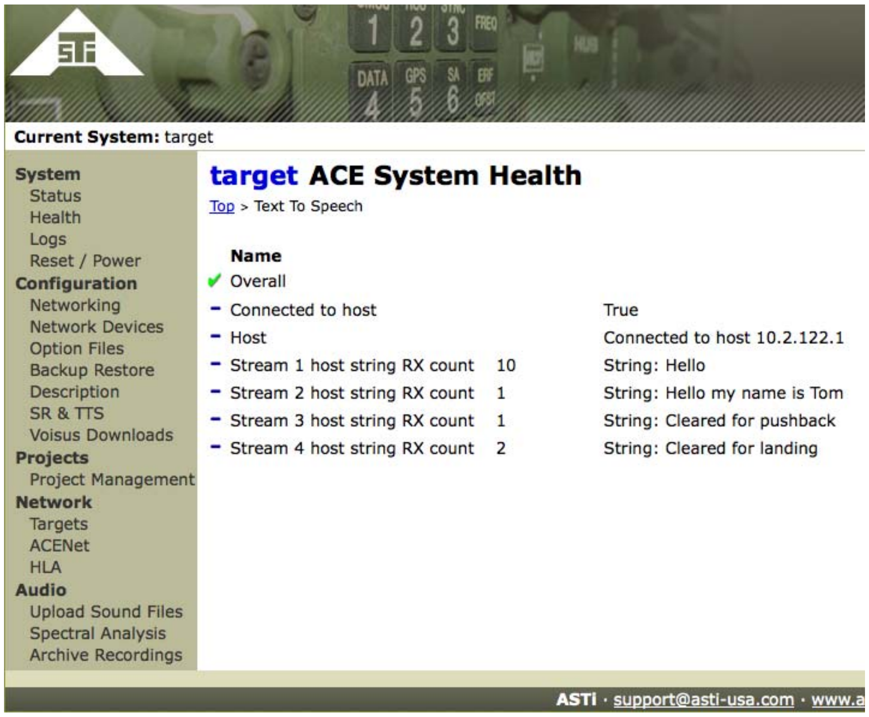
```
Connected to TTS daemon

started -m "Hello my name is Tom " -s 1 --delay="0" --rate="5" --volume="8" --
language="American English" --voice="Tom"

finished -m "Hello my name is Tom " -s 1 --delay="0" --rate="5" --volume="8" --
language="American English" --voice="Tom"
```

RMS

Some debugging information is available for TTS in the RMS Health System under the “Text To Speech” link. This page reports:

- Host connection status
- Host IP address
- Number of commands received for each stream
- Last speech string for each stream
- Any errors encountered on a stream

Take note of any red X's as these may indicate a problem with licensing or a host command. If more streams are used in the model than are available in the ASTi options file or TTS license file an error is displayed. Host command errors include setting an invalid rate, specifying a voice that hasn't been installed on the Target, and others.



target ACE System Health

Top > Text To Speech

	Name		
✓	Overall		
−	Connected to host		True
−	Host		Connected to host 10.2.122.1
−	Stream 1 host string RX count	10	String: Hello
−	Stream 2 host string RX count	1	String: Hello my name is Tom
−	Stream 3 host string RX count	1	String: Cleared for pushback
−	Stream 4 host string RX count	2	String: Cleared for landing

Command Syntax and Parameters

The TTS system expects commands from the host in a simple text format. The syntax is the same as the one that is used to pass options to any program in a Linux terminal window. The host should send one command per TCP packet to the TTS system and end each command with a new line character. The order of the options does not matter.

The available parameter options are as follows. The only option that is required to be in every command is -s or --stream.

Command Options

Options	Descriptions
-m MESSAGE, --message=MESSAGE	Text to be converted into speech.
-d DELAY, --delay=DELAY	Delay in seconds between loop repetitions.
-p, --playall	Prevents the speech from being interrupted.
-s STREAM, --stream=STREAM	The stream number this command applies to. If not specified, the command will have no effect. Specify the option more than once to apply it to multiple streams. This should correspond to the stream number set in the Text-to-Speech component in the model.
-r RATE, --rate=RATE	Specify the rate or cadence of the voice as an integer between 1 and 9. Default is 5.
-v VOLUME, --volume=VOLUME	Specify the volume of the voice as an integer between 0 and 9.
-l LANGUAGE, --language=LANGUAGE	Specify the voice's language. Usually a voice has only one language paired with it, so this option is unneces-sary.
-w VOICE, --voice=VOICE	Specify the voice.
-i, --interrupt	Stop the currently speaking voice. Also clears any other queued commands.
-c, --continuous	Loop the message continuously.
-t, --tag	Specify a text tag that will be referenced in the response to the host.

Command Behavior

The TTS system queues commands received from the host, processing them one at a time in a first in, first out (FIFO) fashion. More specifically, each TTS stream has its own queue and operates independently of the others. Each stream extracts a command from its queue, sets the various parameters contained within (voice, rate, etc.) and then begins generating speech. When the speech is finished, the next command is started.

Several parameters are available that affect the behavior. If speech should loop repeatedly, use the continuous -c option. If a command should preempt or stop speech from a previous command (including looping speech), use the interrupt -i option. The interrupt option also clears the entire queue of commands for that stream. If speech should not be interrupted by a new command, even if the -i option is present, use the playall -p option. For looping speech, the delay -d option can be used to insert silence between loops. Most message sequencing requirements can be achieved using a combination of these options.

Responses Sent to the Host

Commands are acknowledged with a text response sent back over the TCP socket to the host. Currently, there are three types of response messages - “speech started”, “speech finished”, and “error”. Responses are sent asynchronously, some time after the command is received. For example, if the interrupt -i option wasn't specified, the current command won't start until speech from the previous finishes. This means the “speech started” message could possibly arrive multiple seconds after the current command was sent.

If a command begins executing (meaning its speech begins), the TTS system will send both “speech started” and “speech finished” messages to the host at the appropriate time. The timing of the started and finished messages is not guaranteed to be exact due to audio buffering in the system. It is possible that a command never begins, in which case the host will not receive a response for it. This could happen if the command was cleared from the queue by a newer command containing the interrupt -i option. For a looping message, started and finished messages are only sent once, not every time through the loop. The finished message is only sent when the loop is interrupted by another command.

If an error is encountered when executing the command, a simple error response is sent. The response will contain the parameter or option that caused the error and possibly some additional debugging information.

It is recommended to use the tag -t option for all commands. If used, it provides an easy way for the host to figure out which command a response message is referring to. The tag will be prepended to the front of the response message. The simplest way to use the tag is to use a number that increments with every command. If the tag -t option is not used, the entire command message is appended to the response instead, making the response useful for only debugging purposes. A different convention could be used, but the one restriction is the tag must be a single word and not contain white space characters.

Examples

Command	-m “Hello my name is Tom” -s 1 -r 5 -w Tom -t 1
Meaning	Say “Hello my name is Tom” on stream 1, with rate 5, using the Tom voice. Send a response with a tag of “1”.
Response	“1 started”, then “1 finished”

Command	-t 2 -s 3 -v 1 -d 2 -c -m “This is a looping message”
Meaning	Say ”This is a looping message” repeatedly on stream 3 at a low volume. Wait 2 seconds between loops. Send a response with a tag of “2”.
Response	“2 started”, then “2 finished”

Command	-t 3 -s 2 -r 99 -m “Cleared for pushback”
Meaning	This command will fail because the rate of 99 is invalid.
Response	“3 error rate”

Command	-t 4 -r 5 -m “Cleared for pushback”
Meaning	This command will fail because no stream was specified.
Response	“4 error stream no stream specified”

4.13. Vibration Analysis

This tool is used to analyze vibrations in the sound field and produce a Power Spectral Density (PSD) plot. The basic method is to record an audio signal in the model, then post-process the recording using a script on the Target.

Note: In order to use the Vibration Analysis tool, the options file must be enabled for this feature.

Step 1: Record the Input Signal for Analysis – see the VibrationCapture component in the ACE Studio Components Reference Guide (DOC-01-TELAS-CRG-4) for more information.

1. Link the AudioIn from an ACU channel to SignalIn of a VibrationCapture component.
2. Set the Trigger to TRUE in order to begin recording. The component will report back once it has recorded thirty seconds of audio.
3. In order to achieve proper scaling on the Y axis of the PSD, change the gain of the Vibration Capture component. For example, if 1V is equal to 1G on the accelerometer, and an ACU channel with no input gain is used for the signal, the gain should be set to 0.4 in the component.

This is a result of a 2.5 V peak to peak value being converted to a 1 to -1 scale inside the ACE modeling environment.

Step 2: Run Vibration Script to Process Data

1. The recorded audio files are located on the Target under:

   `/var/local/asti/recordreplay/`

2. The recorded audio files are named with a paradigm '`library000_group65535_index###.tsr`' where the index number is set within the VibrationCapture component.
3. The Vibration Analysis script can be run from the command line of the Target. The command requires several options and parameters in order to run:

   -i, --input=</path/to/file>: data file to process

   -o, --output=</path/to/file>: output PDF file name

   -s, --window_size=#: size of the FFT window in seconds

   -l, --window_overlap=#: FFT window overlap percentage as a decimal, e.g. 50 % = .5

   -t, --window_type=#: Type of FFT window to apply, choices are: 0=Hann, 1=Rectangle, 2=Bartlett

   -f, --downsample_freq=#: frequency to downsample to, choices are: 60, 100 (default=60)

   One of the key parameters is window size, which determines the resolution of the final PSD plot. For example, a window size of 1.5 seconds, with a down sample frequency of 100Hz will produce 150 data points between 0 and 50Hz, giving a resolution of 0.333Hz.

4. It is recommended that the script be run as root to prevent permissions errors. Command Usage:

   ```
   vibration_graphgen.py -i <InputFileName> -o <OutputFileName>
   -s <WindowSize> -l <WindowOverlapPercentage> -t <WindowType>
   -f <OutputSamplingRate>
   ```

   Example:

   ```
   vibration_graphgen.py -i /usr/local/asti/recordreplay/
   library000_group65535_index001.tsr -o /home/admin/
   results.pdf -s 10 -l 0.75 -t 0 -f 60
   ```

   Note that the Power Spectral Density plot is always a .pdf file and must be named so in the command.

5. Two additional files are generated, rawOut<date>.txt and PSDOut<date>.txt. These files are the raw audio recorded during the test and the Power Spectral Density plot values. Both files are newline delimited lists of float32 values. These files are generated in the same directory where the script runs.

4.14. Voisus Client

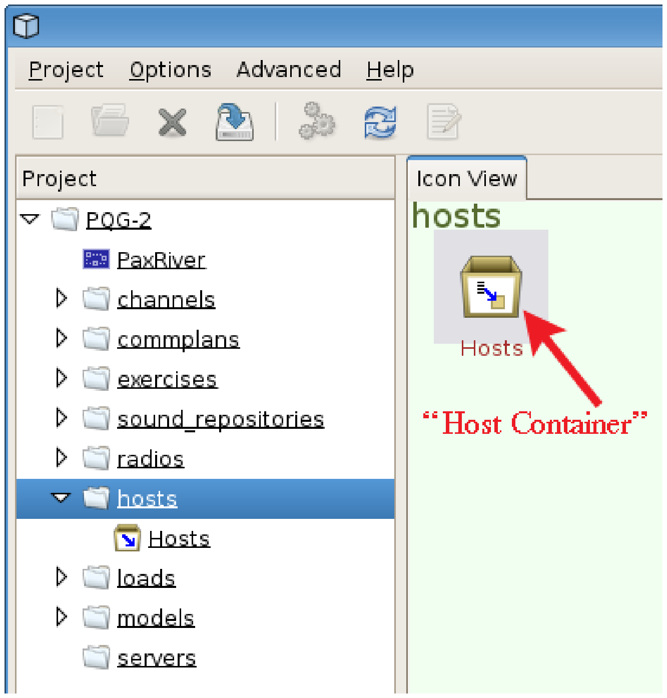
ASTi’s Voisus client-server technology offers an economical solution for network voice communications, simulation and audio distribution. Voisus technology allows users to deploy Voisus operators on existing workstations (clients), providing a convenient, economical and highly scalable network communications solution with ASTi’s simulated radios and intercoms hosted on the Target (server).

Voisus client is a communication operator GUI panel with remote IP audio. Clients are set up in Studio using the Channel Helper.

Voisus clients are for use with ASTi’s Synapse Voisus Server (P/N: SYN4-VS-XX). See product documentation, Synapse Voisus Server Manual (DOC-01-SYN4-VS-1) for system overview, configuration and requirements.

5.0. Host Interface

The ACE host interface allows state data to be passed between a user's host application software and the ACE sound and communications loads running on the Targets. State data is transferred via network packets using User Datagram Protocol (UDP).



The ACE Studio host interface is made up of two parts, host containers and their corresponding sockets at the project level, and host I/O packets at the model level.

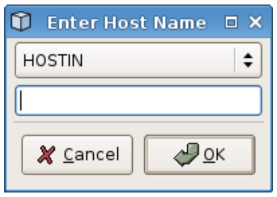
Each host container is used to configure the packet interface with the appropriate UDP port number and physical Ethernet port (eth0, eth2) on a Target.

In the host model, each host I/O packet is used to define the information contained in the Host UDP packet. The host I/O packet is commonly called the Interface Control Document (ICD).

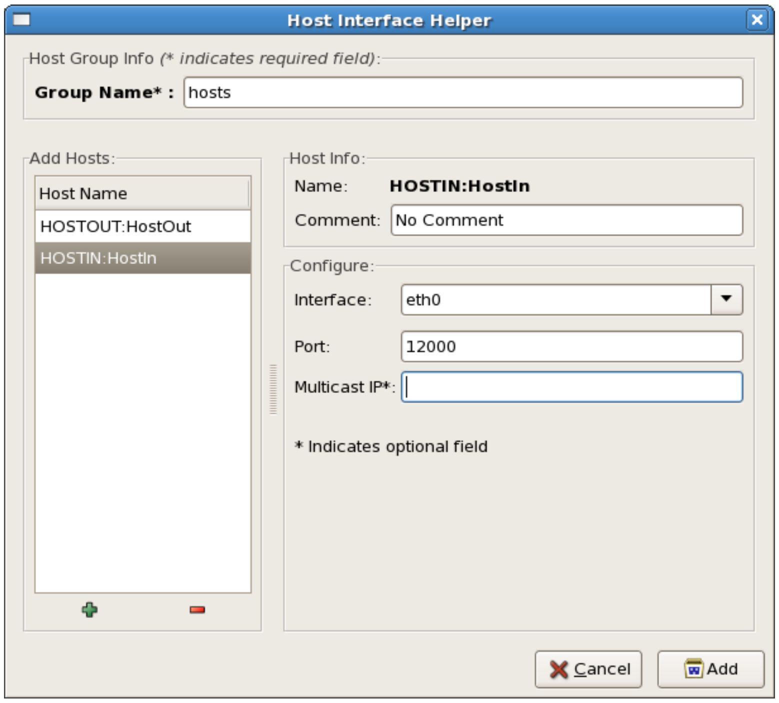
This approach dereferences the models such that they carry no specific network configuration information, making them reusable across platforms without configuration changes.

5.1. Host Control in the Project Level

In Project > Host the user adds the host containers and sockets are created within each host container. Each socket is defined as either HostIn or HostOut.



For each HostIn socket created, the user must define the interface and port number. The port number selects the default network receive port for the packet data if it is an input packet or the transmit network port if it is an output packet. The multicast IP is optional.



Host Group Info (* indicates required field):

Group Name* : hosts

Add Hosts:

Host Name
HOSTOUT:HostOut
HOSTIN:HostIn

Host Info:

Name: **HOSTIN:HostIn**

Comment: No Comment

Configure:

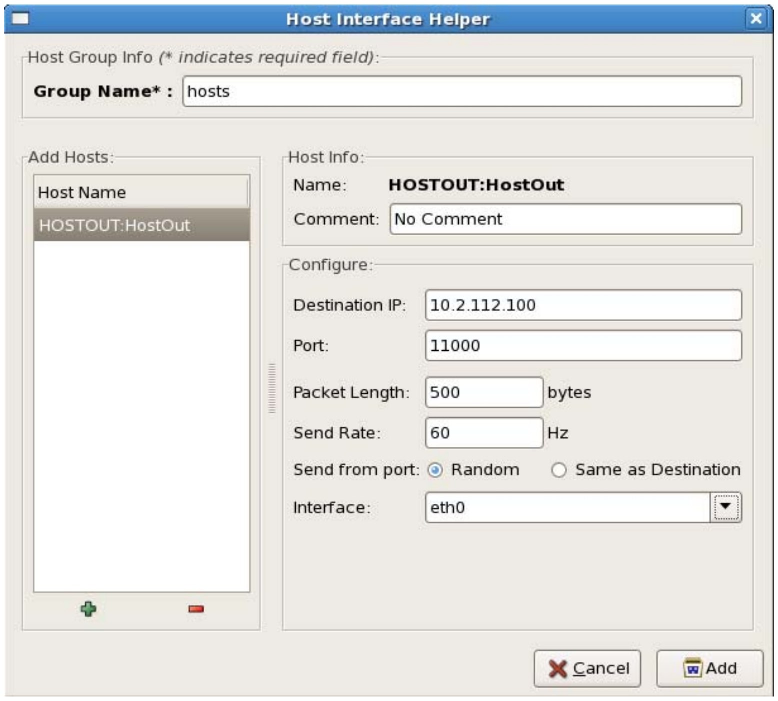
Interface: eth0

Port: 12000

Multicast IP*:

* Indicates optional field

For each HostOut socket created, the user must define the destination IP address, port, packet length in bytes, and the send rate in hertz.



Group Name* : hosts

HOSTOUT:HostOut

Name: **HOSTOUT:HostOut**

Comment: No Comment

Destination IP: 10.2.112.100

Port: 11000

Packet Length: 500 bytes

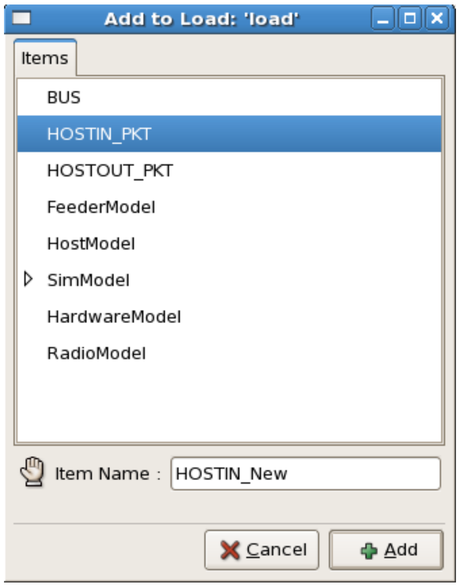
Send Rate: 60 Hz

Send from port: [✓ Random] Same as Destination

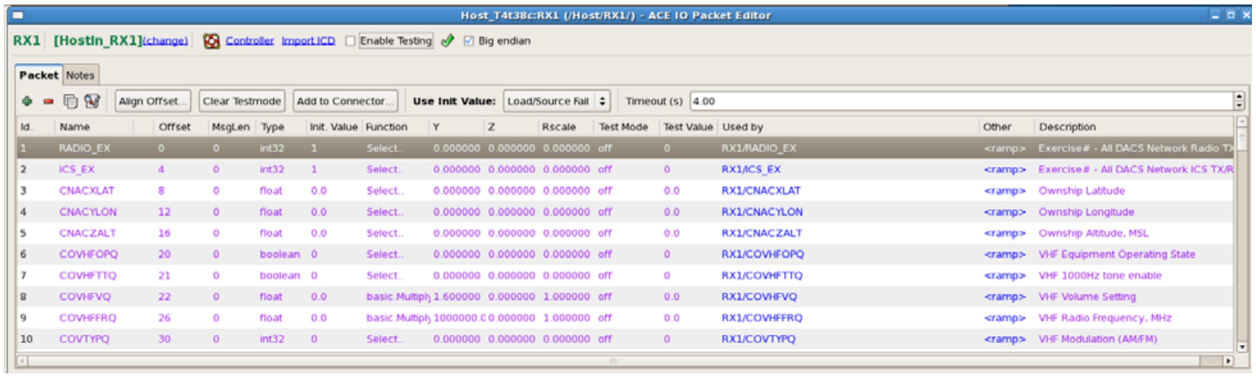
Interface: eth0

5.2. Host Control in the Load and Model Level

In the Load, add a host model. This is required in order to create the host I/O packets. Inside the host model, select to add either a HostIn or HostOut packet.



Before getting into the details of the HostIn and HostOut packets, there are a few general things to note about creating packets. Each packet must be assigned to a socket, only one host I/O packet can be linked to a single socket at a time. Select the '**change**' link to assign the packet to a socket.



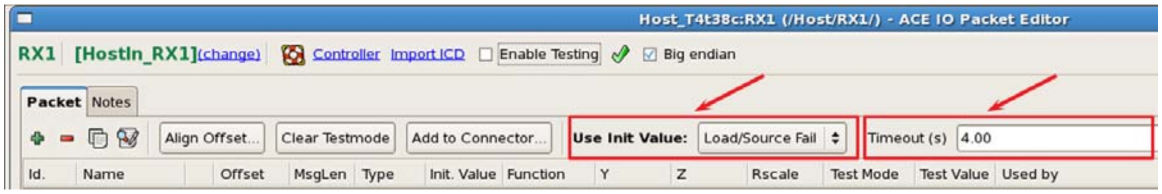
Use Init Value: Load/Source Fail

Timeout (s) 4.00

Id.	Name	Offset	MsgLen	Type	Init. Value	Function	Y	Z	Rscale	Test Mode	Test Value	Used by	Other	Description
1	RADIO_EX	0	0	int32	1	Select..	0.000000	0.000000	0.000000	off	0	RX1/RADIO_EX	<ramp>	Exercise# - All DACS Network Radio TX
2	ICS_EX	4	0	int32	1	Select..	0.000000	0.000000	0.000000	off	0	RX1/ICS_EX	<ramp>	Exercise# - All DACS Network ICS TX/R
3	CNACXLAT	8	0	float	0.0	Select..	0.000000	0.000000	0.000000	off	0.0	RX1/CNACXLAT	<ramp>	Ownship Latitude
4	CNACYLON	12	0	float	0.0	Select..	0.000000	0.000000	0.000000	off	0.0	RX1/CNACYLON	<ramp>	Ownship Longitude
5	CNACZALT	16	0	float	0.0	Select..	0.000000	0.000000	0.000000	off	0.0	RX1/CNACZALT	<ramp>	Ownship Altitude, MSL
6	COVHFOPQ	20	0	boolean	0	Select..	0.000000	0.000000	0.000000	off	0	RX1/COVHFOPQ	<ramp>	VHF Equipment Operating State
7	COVHFTTQ	21	0	boolean	0	Select..	0.000000	0.000000	0.000000	off	0	RX1/COVHFTTQ	<ramp>	VHF 1000Hz tone enable
8	COVHFVQ	22	0	float	0.0	basic.Multiply	1.600000	0.000000	1.000000	off	0.0	RX1/COVHFVQ	<ramp>	VHF Volume Setting
9	COVHFFRQ	26	0	float	0.0	basic.Multiply	1000000.0	0.000000	1.000000	off	0.0	RX1/COVHFFRQ	<ramp>	VHF Radio Frequency, MHz
10	COVTYPQ	30	0	int32	0	Select..	0.000000	0.000000	0.000000	off	0	RX1/COVTYPQ	<ramp>	VHF Modulation (AM/FM)

Set the ‘**timeout**’ value. The timeout value is the amount of time that passes without activity from the host, i.e., the time since the last packet was received. This value must be an integer value between 0-99. If a timeout occurs and the initial value is not set to Load/Sourcefail then it will maintain the previous values.

Select an option from the ‘**Use Init Value**’ drop down list to set when the initial value is used based on the timeout. The options include never, load, and load/sourcefail. Load is used initially upon loading the Project. Load/sourcefail retrieves the values upon load installation unless the host fails (i.e. times out) and then it returns back to the initial values.



Check the ‘**Enable Testing**’ box to toggle the test mode on and off for testing the host control.

Press the ‘**Live Capture**’ (green check) to capture the live values coming in from the host.

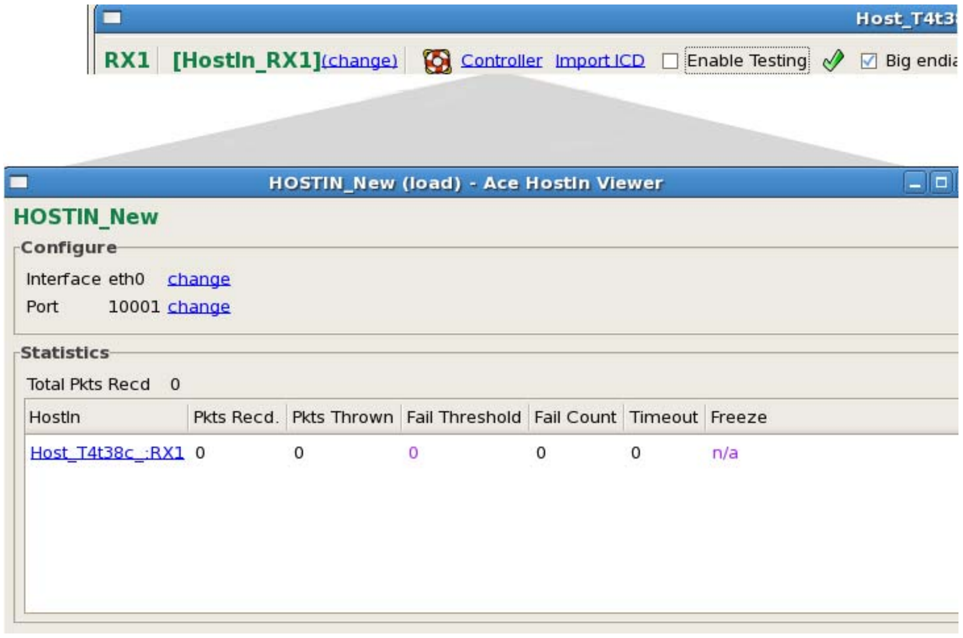
Select ‘**Big Endian**’ to change the byte order. The endianness defines the byte order for the data in a packet.

Select ‘**Align Offset**’ to auto-align the selected variables.

Select ‘**Clear Testmode**’ to clear the test values.

Select ‘**Add to Connector...**’ to add all selected variables to a connector.

Select the '**Controller**' link to view the socket and packet statistics. For HostIn packets, there is a column for the '**Fail Threshold**' time. If a packet has not arrived within the set amount of time then a communication failure is assumed. The '**Freeze**' field allows you to ignore the incoming packets and they are essentially thrown away. If you change these fields and then perform a save and reinstall the layout, the values are lost.



HOSTIN_New

Configure

Interface eth0 change

Port 10001 change

Statistics

Total Pkts Recd 0

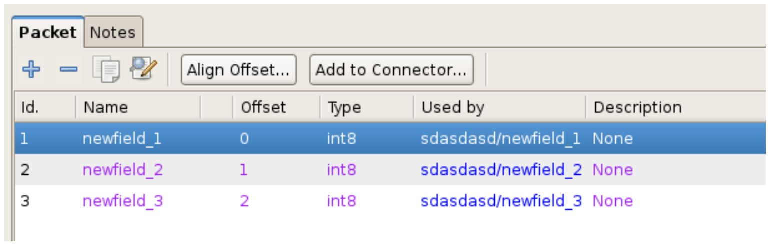
HostIn	Pkts Recd.	Pkts Thrown	Fail Threshold	Fail Count	Timeout	Freeze
Host_T4t38c_:RX1	0	0	0	0	0	n/a

For HostIn the host I/O packet is used to define the values for the information coming in on the port. The IO packet values include:

- **Name** – Enter the HostIn variable name.
- **Offset** – Sets the offset location in the Ethernet packet for the data associated with the variable.
- **Type** – Sets the variable type and data type for the variable.
- **Init. Value** – Sets the initial value for the variable. The '**Use Init Value**' drop-down list sets when the value will be used. The options include never, start the load, and end of the load.
- **Function** – Adds a math function to apply to the variable.
- **Scaler** – Adds a scale factor to apply to the gain output of the function.
- **Test Mode** – Toggles between using the host value or the value set in the Test Value column.
- **Test Value** – Sets the value used for overriding the host value.
- **Used by** – Sets where the variable is being sent.
- **Other** – <ramp> Ramps the test values up or down.
- **Description** – Add details about the variable.

For HostOut, the host I/O packet is used to define the values for the information going out on the port. The IO packet values include:

- **Name** – Enter the HostOut variable name.
- **Offset** – Sets the offset location in the Ethernet packet for the data associated with the variable.
- **Type** – Sets the variable type and data type for the variable.
- **Used By** – Sets where the variable is coming from.
- **Description** – Add details about the variable.



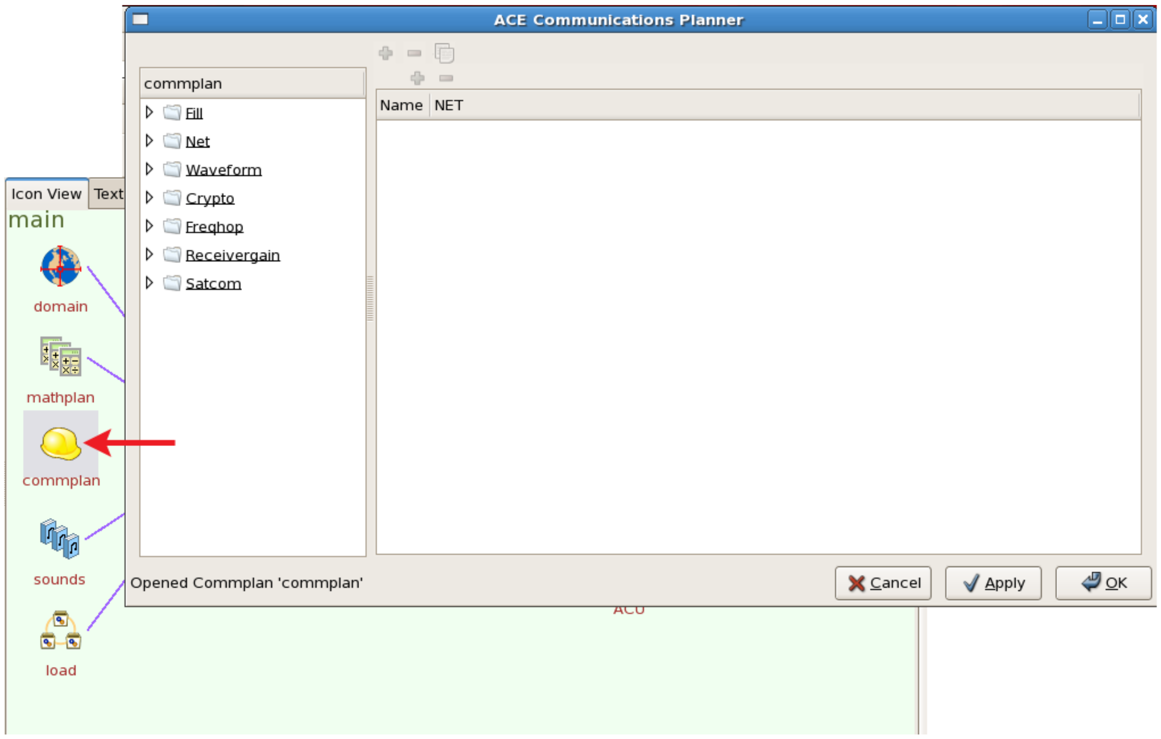
Id.	Name	Offset	Type	Used by	Description
1	newfield_1	0	int8	sdasdasd/newfield_1	None
2	newfield_2	1	int8	sdasdasd/newfield_2	None
3	newfield_3	2	int8	sdasdasd/newfield_3	None

6.0. Comm Plans

The Comm Plan tool provides the defining parameters for the configuration of an ASTi simulated communications system. The software generated files come with library presets or users can create new radio parameters to meet their specific needs.

There are a few key concepts that must be understood before developing a Comm Plan.

- **Fill –** Contains the full set of parameters known as the Comm Plan.
- **Net –** Each Net includes the common parameters that networked radios must share for successful inter-communications. Net parameters include: frequency, Tx frequency, waveform, crypto settings and frequency hopping settings.
- **Waveform –** Includes the modulation type, type of encoding for the wave, rate, bandwidth, transmitter power, and receiver gain.
- **Crypto –** Set the crypto key and the crypto system for secure radio transmission.
- **Frequency Hopping –** Sets the simulation settings for HAVE QUICK and SINCGARS frequency hopping effects through simulated noise resistance and time drifting effects. Frequency Hopping includes the following parameters: net ID, Hopset Wod, Lockout ID, Sync Tod, and Transec key.
- **Receiver Gain –** Adjusts the strength of the radio. Receiver Gain includes the following parameters: sensitivity (dB), bandwidth overlap, AGC Target (dBm), AGC Gain Limit (dB), and AGC Ramp Factor.
- **Satcom –** This will be implemented in a future software version.

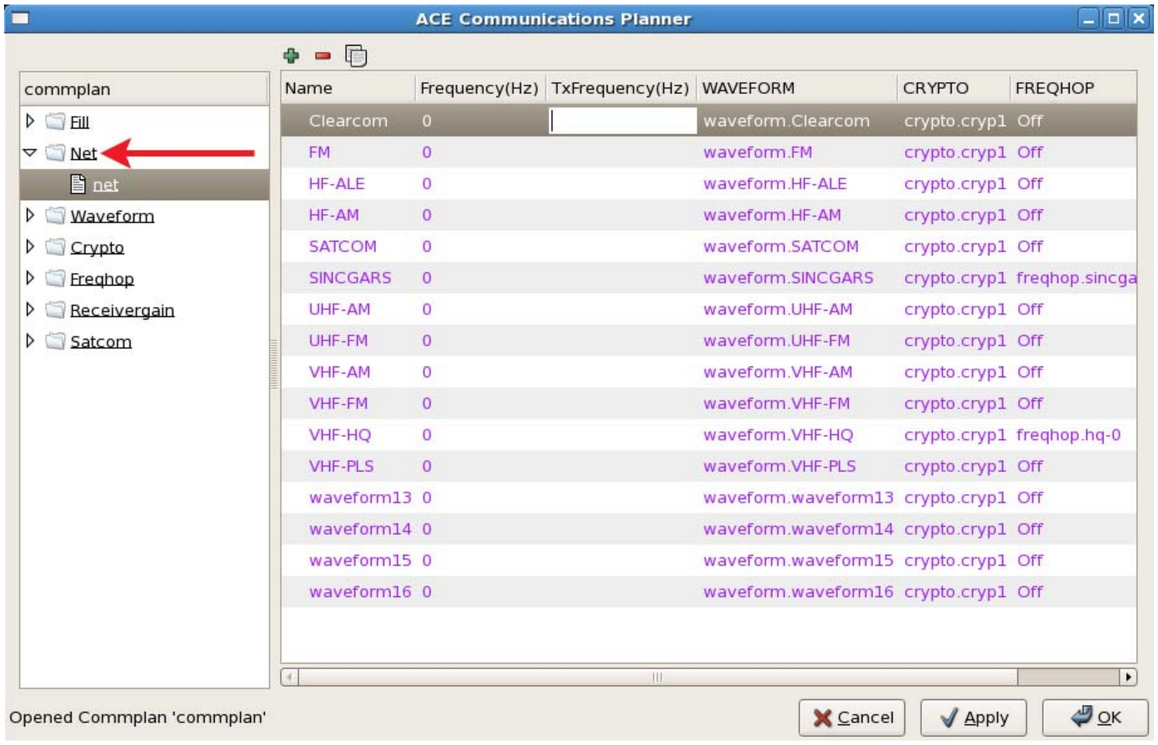


Opened Commplan 'commplan'

During Comm Plan building sessions, the user can reuse the library presets, modify existing presets or add new sets of parameters. Building Comm Plans will become much quicker and easier as the user builds up the component libraries.

6.1. Building Libraries

The library items can be changed or created at any time or in any order. However, ASTi recommends that you build the Comm Plan libraries by starting with receiver gain and working your way up the list.



Name	Frequency(Hz)	TxFrequency(Hz)	WAVEFORM	CRYPTO	FREQHOP
Clearcom	0		waveform.Clearcom	crypto.cryp1	Off
FM	0		waveform.FM	crypto.cryp1	Off
HF-ALE	0		waveform.HF-ALE	crypto.cryp1	Off
HF-AM	0		waveform.HF-AM	crypto.cryp1	Off
SATCOM	0		waveform.SATCOM	crypto.cryp1	Off
SINCGARS	0		waveform.SINCGARS	crypto.cryp1	freqhop.sincga
UHF-AM	0		waveform.UHF-AM	crypto.cryp1	Off
UHF-FM	0		waveform.UHF-FM	crypto.cryp1	Off
VHF-AM	0		waveform.VHF-AM	crypto.cryp1	Off
VHF-FM	0		waveform.VHF-FM	crypto.cryp1	Off
VHF-HQ	0		waveform.VHF-HQ	crypto.cryp1	freqhop.hq-0
VHF-PLS	0		waveform.VHF-PLS	crypto.cryp1	Off
waveform13	0		waveform.waveform13	crypto.cryp1	Off
waveform14	0		waveform.waveform14	crypto.cryp1	Off
waveform15	0		waveform.waveform15	crypto.cryp1	Off
waveform16	0		waveform.waveform16	crypto.cryp1	Off

Opened Commplan 'commplan'

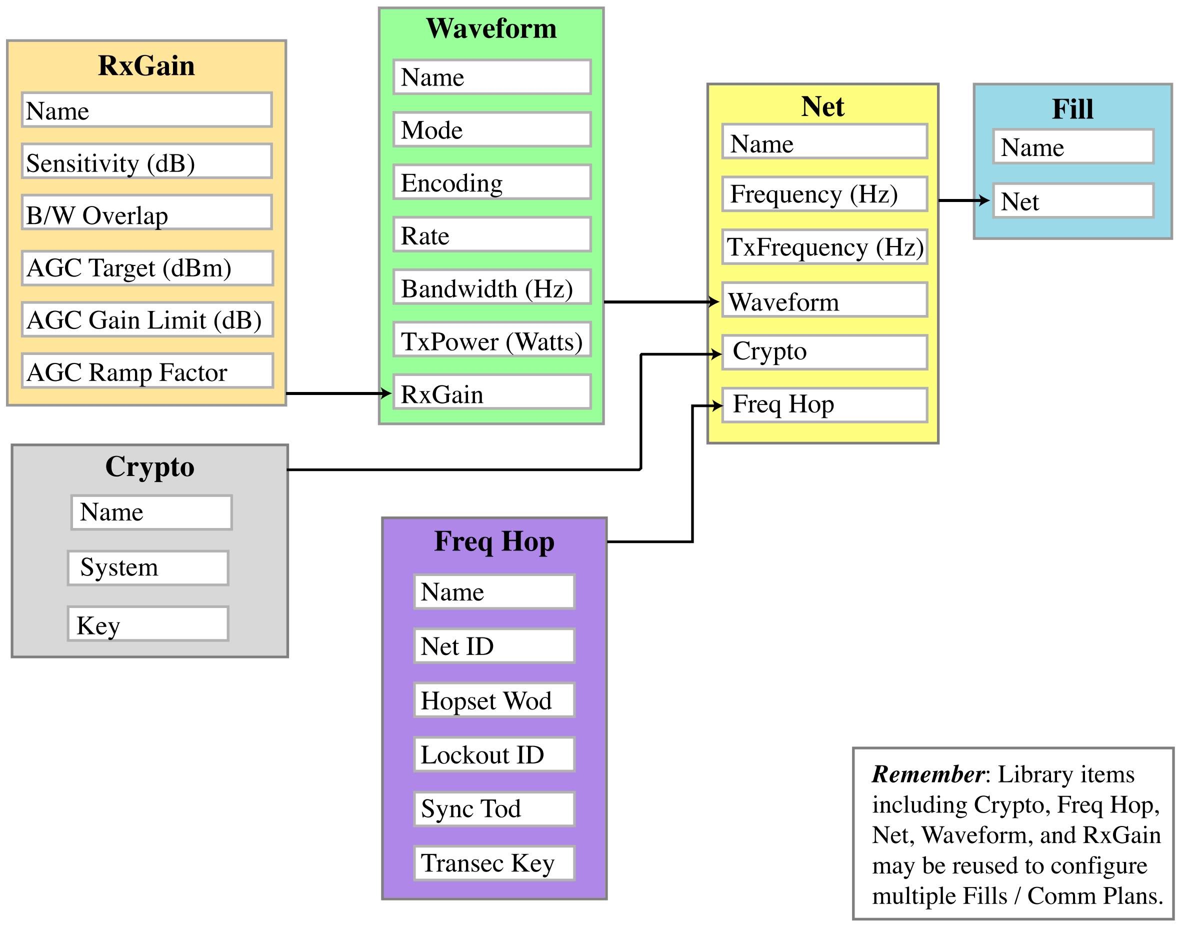


Figure 5: Comm Plan Functional Diagram

7.0. ACE Server Applications

The Target hosts a powerful suite of server-based applications, used to further enhance the simulated communications environment, extend communications and modeling capabilities, and provide protocol translation and interfacing. Applications run on the Target platform and provide enhanced capabilities for one to many Target platforms.

Server features include:

- **HF**: High-fidelity high-frequency (HF) propagation
- **SATCOM**: DASA, DAMA, uplink/downlink, delay, encryption
- **Terrain**: World database, DTED/DEM, and fresnel zone/diffraction
- **Link 16**: Terminal simulation propagation, NPG, data rate simulation, stacked/crypto nets, SISO Tadil-Tales support
- **HLA**: Multiple RTI vendor/version, DDM, connectionless mode, and multiple federation support, NASMP and DMO compliant, remote federation management (see HLA section in this document for more information).
- **State Modeling**: Hosting of system-wide state models (e.g. dial-up/pbx applications)
- **Protocol Translation/Interfacing**: e.g. VoIP, non-standard simulations/protocols

7.1. HF Propagation

ACE Studio provides real-time, high-fidelity modeling of HF radios in the radio environment. The software computes the propagation effects between virtual radios, taking into account things such as transmitter-receiver global position, season, time of day (day-night terminator), and solar activity.

To properly simulate solar activity and seasonal/circadian effects on the ionosphere and HF radio signal propagation a Smoothed Sunspot Number (SSN) and Time-of-Day offset is required.

The time-of-day offset sets the simulation day of year and time of day. It is expressed as an offset, in hours, between the system clock (local time in GMT) and the simulation time in GMT.

7.2. Terrain

7.2.1. Terrain Interface

ACE Terrain interface software applies occulting and degradation effects to communication paths in the radio environment. All terrain and propagation effects are set in the Comm Plan Receiver Gain section.

For more information on simulating the terrain effects please see ASTi Application Note 94: Setting Path Loss Effects in T4 ACE (www.asti-usa.com/support/appnotes/94.html).

7.2.2. Terrain Server

The Telestra/ ACE Terrain Database Server software features include:

- Ability to serve multiple ASTi radio models in multiple exercises/ federations simultaneously
- Support for terrain data in Digital Terrain Elevation Data (DTED)
- Pre-installed DTED level 0 terrain data
- Ability to add customer-provided terrain data
- Optimized path loss calculations

See the Target Cold Start Procedure (DOC-02-TEL4-TCS-1) section 7.0. “Installing the Telestra 4 Terrain Server Software” for more information.

To use third party terrain data, navigate to RMS using a standard web browser. Under the Configuration section select the “Terrain” tab. Follow the instructions to activate the third party terrain data.

8.0. HLA

8.1. HLA Introduction

Unlike many other HLA solutions, ASTi's HLA implementation was developed from the ground up to fully exploit the flexibility and interoperability envisioned under DMSO's High Level Architecture (HLA 1.3) standard. Multiple RTI support, established and published Radio SOM, agile FOM capabilities, backchannel communications options, and debug tools offer users a well supported HLA environment. In addition, ASTi's Target takes advantage of high performance, industrial, off-the-shelf technology to provide increased HLA performance and reliability. The HLA software is highly flexible with capabilities to support HLA operation for radio and communications models.

However, with this flexibility comes additional complexity. The first thing to remember about HLA is that it is an architecture and not a standard. Unlike DIS where the formatting of the transmitter and signal PDUs is fixed, HLA gives the user the flexibility to define the structure of each object and interaction on the network. The downside of this flexibility is that the implementation is rarely (if ever) a plug and play system. For example, when you look at the objects and interactions related to audio, you will see there are numerous approaches to the problem. There are too many complex components such as the RTI version, RID file, FED file, convert file, etc. that all affect the final on-wire data structure to assume that setup of an HLA-compliant trainer is as simple as its DIS counterpart.

8.2. Overview of RTI and RID Files

In computing, run-time infrastructure (RTI) is a middleware that is required when implementing the High Level Architecture. RTI is the fundamental component of HLA. It provides a set of software services that are necessary to support federates to coordinate their operations and data exchange during a runtime execution. Each RTI has an associated RTI Initialization Data (RID) file that is used to set up RTI-specific initialization parameters.

ASTi’s Target platform has been tested and supports versions of the VTC RTIs. ASTi is involved with an ever growing number of HLA-based communications simulations for a variety of programs throughout the US and internationally. The RTI versions currently tested and supported vary based on ACE software release. For details on the specific supported RTI versions please see Telestra 4 Target Operation and Maintenance Manual (DOC-01-TEL4-TUG-1), Appendix C: HLA RTIs Compatibility.

8.3. Overview of Federation and Convert File

The audio communications interface to the HLA environment is defined through a number of configuration files on the Target, mainly the Federation (FED) file and the convert (.conv) file. The FED file is used to supply the RTI with all necessary federation execution details during the creation of a new Federation. The FED file also defines the potential routing spaces, objects, and interactions used during the federation. The ASTi HLA radio environment class names and hierarchies are read from a configuration file, and not compiled into the code, allowing agility in switching from one FOM to another. This configuration file is known as the convert file and resides on the Target. The purpose of the convert file is to:

- Provide the names of the object class attributes and interaction classes to which the Target federate subscribes.
- Direct the Target federate to locate specific data within received attribute updates and interactions.

This allows the ASTi SOM to be included in any FOM (Fed file) and is selected simply by choosing the appropriate convert file. One of the most important lessons learned from our experience with HLA is that SOM and FOM design is not just a matter of deciding what units to use. Issues of efficiency, reliability, and coherence of data are all factors of importance that were discovered during the FOM design phase.

8.4. HLA Configuration

There are two main types of HLA configuration in T4:

1. The single platform solution
2. The two platform solution

The Telestra 4 Target is capable of running the HLA software on the same platform with the communication models including the radios, network intercoms, engines, etc. Alternatively, the HLA software can be configured to run on a separate dedicated Target platform. In this case, the Target is acting as more of a gateway to the HLA world.

This section will first describe the configuration for the single platform solution followed by the two platform solution. Determine the appropriate setup for your system before continuing.

Below are two example configurations.

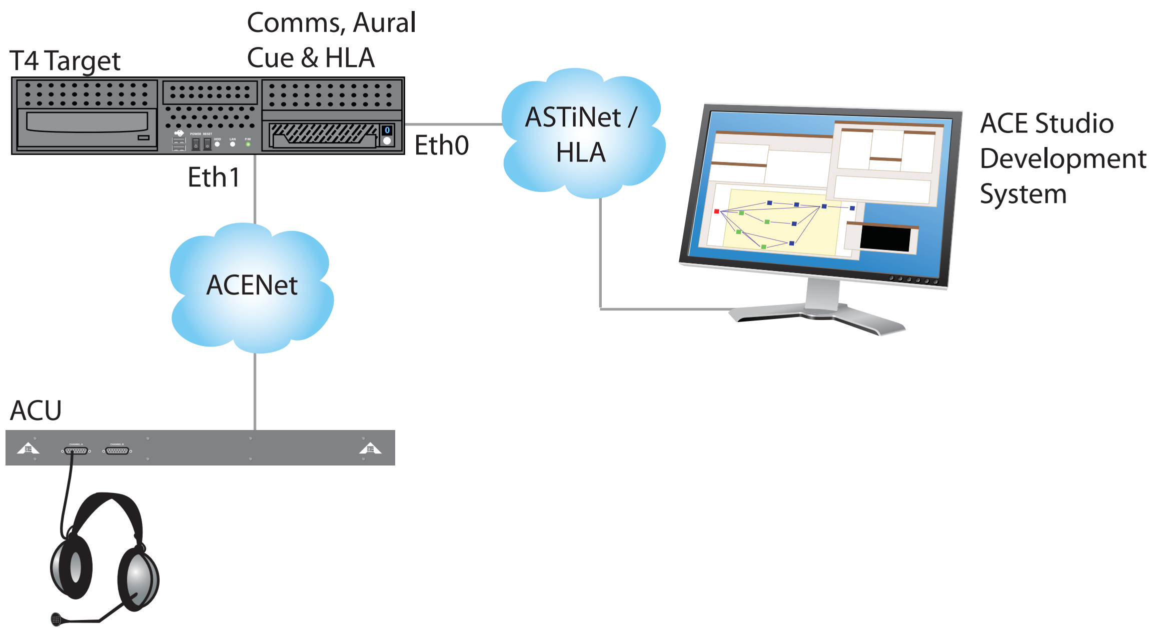


Figure 6: Single Platform Solution

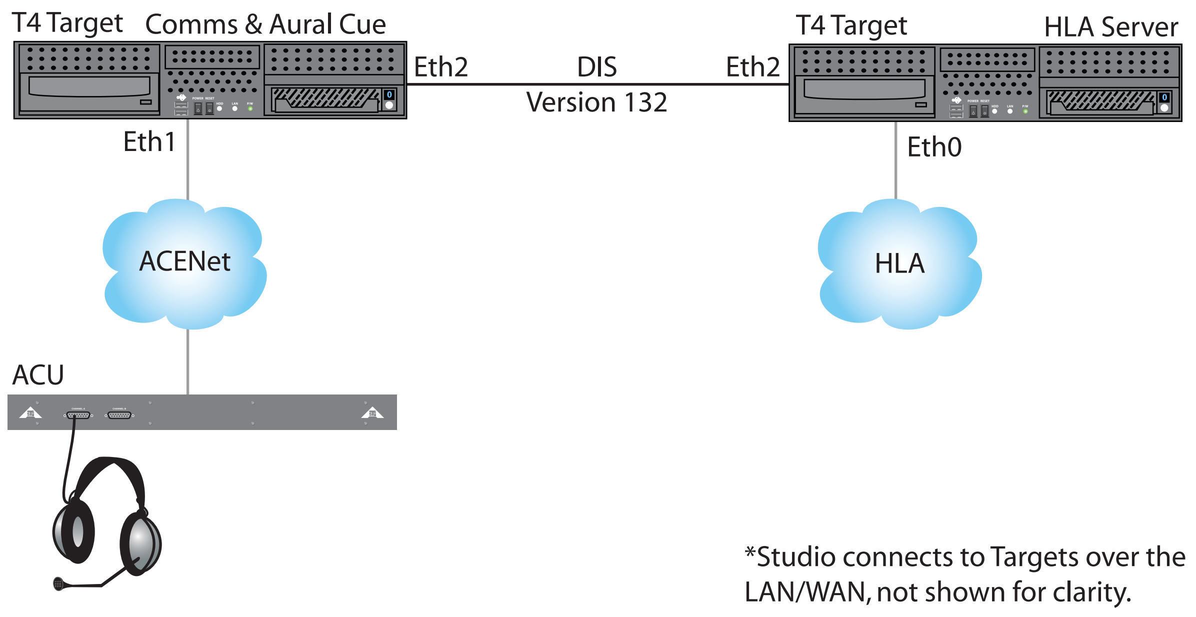


Figure 7: Two Platform Solution

8.5. Single Platform Solution

This solution is supported in ASTi software version 4.16 and above.

The single platform solution combines the communications and aural cue system with the HLA environment. Please continue with section '**8.5.1 Configuring the Target**' for specific details.

8.5.1. Configuring the Target

For this example, the Target is configured for HLA and communications on a single platform.

Step 1: Install and Activate the RTI

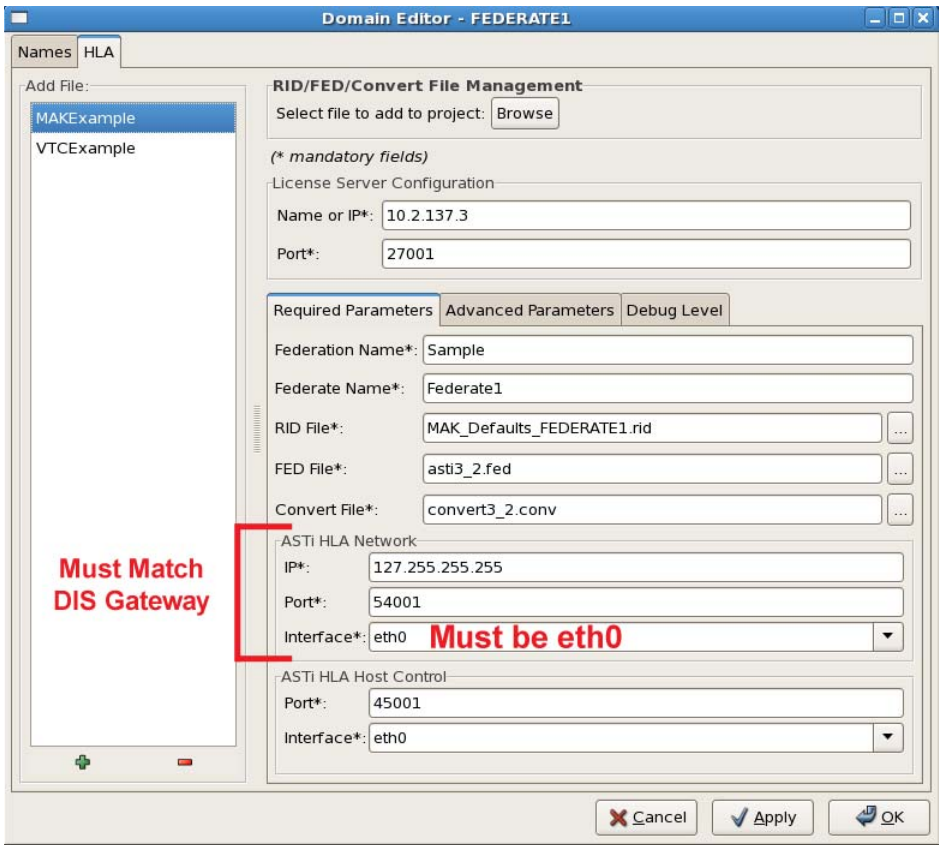
Use RMS to install and activate the RTI. Refer to the Telestra 4 Remote Management System 4 User Guide (DOC-01-TEL4-RMS4-UG-4) for details. You can find this and all ASTi documentation at www.asti-usa.com/support/document.

Step 2: Add a New HLA Configuration File

The HLA Configuration file defines the Federation name, Federate Name, RID File, FED File, etc. Multiple Configuration files can be defined and used by the Target platform. This allows the user to create multiple HLA configurations and switch between them.

Domain Icon

To add a new HLA configuration file open your Domain, select the HLA tab, and select the '+' symbol. Below are examples of standard MAK and VTC configuration files.



RID/FED/Convert File Management
Select file to add to project: Browse

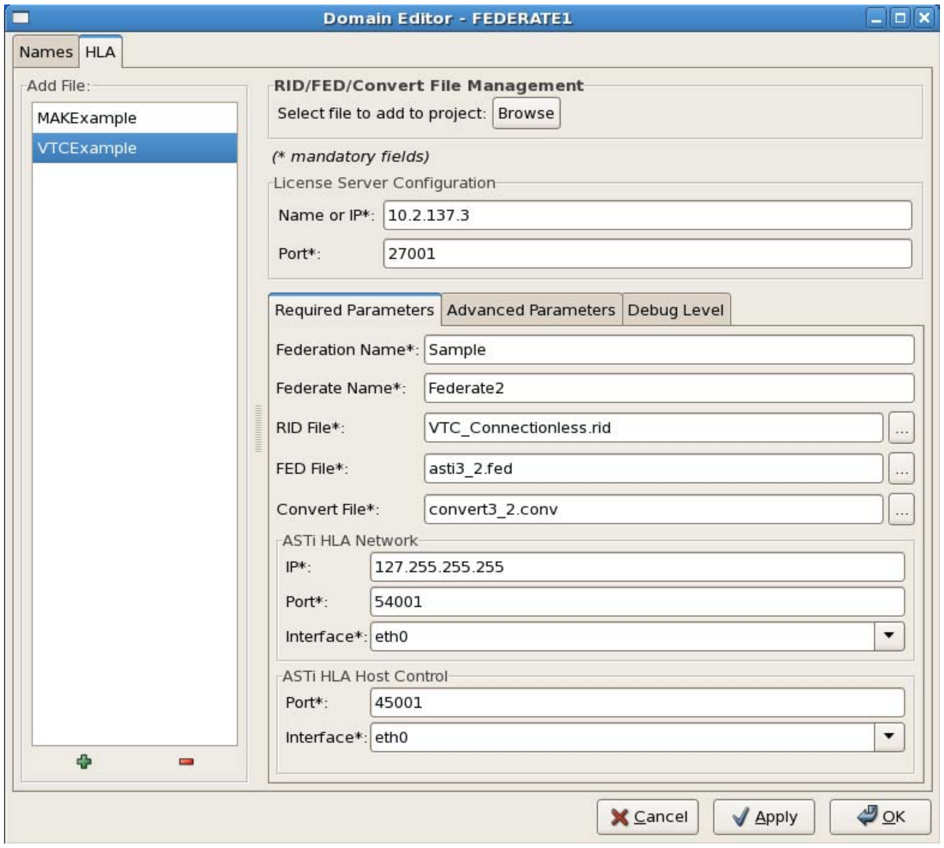
(* mandatory fields)

License Server Configuration
Name or IP*: 10.2.137.3
Port*: 27001

Federation Name*: Sample
Federate Name*: Federate1
RID File*: MAK_Defaults_FEDERATE1.rid
FED File*: asti3_2.fed
Convert File*: convert3_2.conv

ASTi HLA Network
IP*: 127.255.255.255
Port*: 54001
Interface*: eth0

ASTi HLA Host Control
Port*: 45001
Interface*: eth0



Name or IP*: 10.2.137.3
Port*: 27001

Federation Name*: Sample
Federate Name*: Federate2
RID File*: VTC_Connectionless.rid
FED File*: asti3_2.fed
Convert File*: convert3_2.conv

IP*: 127.255.255.255
Port*: 54001
Interface*: eth0

Port*: 45001
Interface*: eth0

Fill in the **Required Parameters**. Parameter definitions are as follows:

License Server Configuration

The license server configuration defines the server IP address or hostname and defines the license server port.

Federation Name

The federation name is the name of the default exercise federation that the Target will be joining.

Federate Name

This is the name of the federate as seen by the federation. Each Target appears as a federate in the federation. The federate name will default to ASTi_Target_1, where Target is the Telestra's hostname. If this federate name is not desired, enter another federate name.

RID File

Select the RID file required for the RTI to operate. Note that you must first upload an RID file to the project by selecting the 'Browse' button from under RID/FED/Convert File management.

FED File

Select the FED file required for the exercise. Note that you must first upload a FED file to the project by selecting the ‘Browse’ button from under RID/FED/Convert File management.

Convert File

Select the convert file required for the exercise. Note that you must first upload a Convert file to the project by selecting the ‘Browse’ button from under RID/FED/Convert File management.

ASTi HLA Network

IP: This must match the DIS gateway. Enter the subnet broadcast IP address as defined by lo (**127.255.255.255**) of the Target.

Port: Enter a port number of **54001**.

Interface: Select **eth0**.

ASTi HLA Host Control

The ASTi HLA Host Control Interface defines which port and interface runs the TCP/IP host control interface. The details of using this interface are defined in the Telestra 4 Target Operation and Maintenance Manual (DOC-01-TEL4-TUG-1).

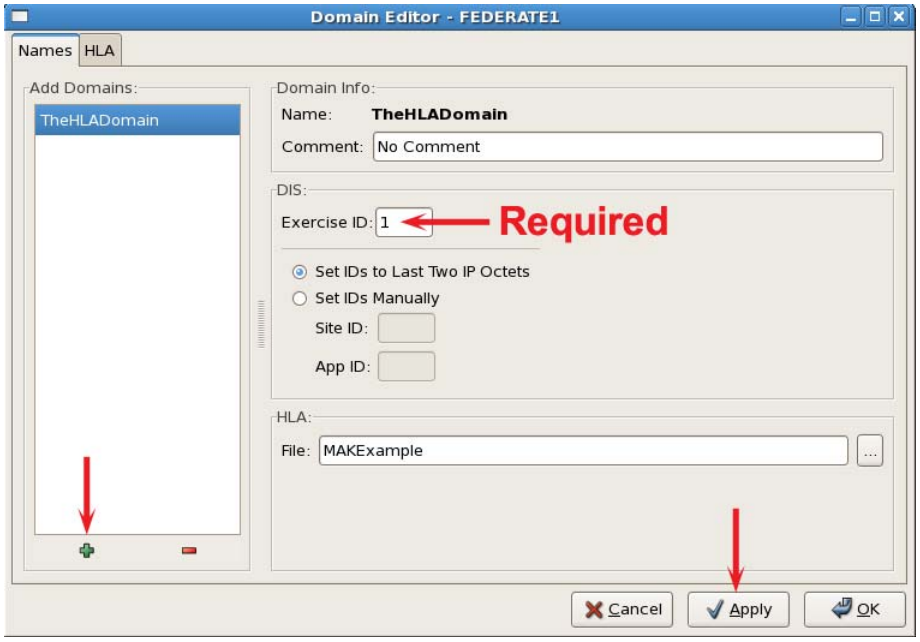
Port: Enter desired port number such as 54001.

Interface: Select the desired interface.

Step 3: Add a New Domain

1. In ACE Studio, navigate to the Domain section using the left menu bar and then double-click the icon.
2. Select the green '+' button to add a new domain.
3. Enter a name for the domain, such as 'TheHLADomain' as shown in the example below.
4. Under HLA, enter the exercise ID. **The exercise ID must be set to 1**.

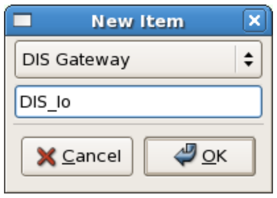
   The Site and Application IDs can be defined here. They set the first two parts of the 64 bit stream tag identifier in the HLA world.
5. Under the 'HLA' add the **HLA Configuration file** that was created in Step 2.
6. Press '**Apply**' to submit the configuration.



Name: TheHLADomain
Comment: No Comment
Exercise ID: 1
[✓ Set IDs to Last Two IP Octets]
Set IDs Manually
Site ID:
App ID:
File: MAKExample

Step 4: Add a DIS Gateway

1. In ACE Studio, navigate to the servers section and right-click on the canvas to add a DIS Gateway.



DIS Gateway

DIS_lo

2. Enter a name such as ‘DIS_lo’ and press ‘**OK**’.
3. Open the DIS Gateway you just created by double-clicking on it. Enter the following:

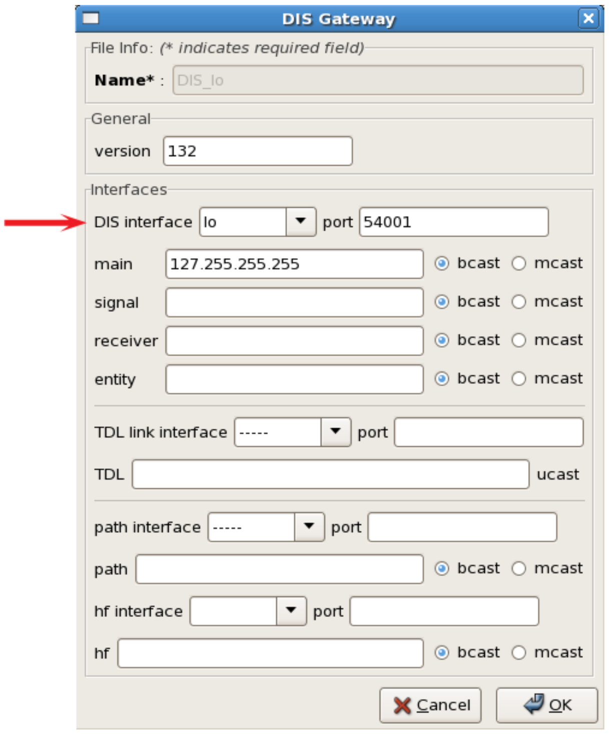
   Version = 132

   > **Note**: Setting the version to 132 is required for the HLA Target to properly process the packets.

   Interface = lo

   Port = 54001

   Main = 127.255.255.255
4. Leave the remaining fields blank and press ‘**OK**’.



File Info: *(* indicates required field)*

Name* : DIS_lo

General

version 132

Interfaces

DIS interface lo port 54001

main 127.255.255.255 [✓ bcast] mcast

signal [✓ bcast] mcast

receiver [✓ bcast] mcast

entity [✓ bcast] mcast

TDL link interface ----- port

TDL ucast

path interface ----- port

path [✓ bcast] mcast

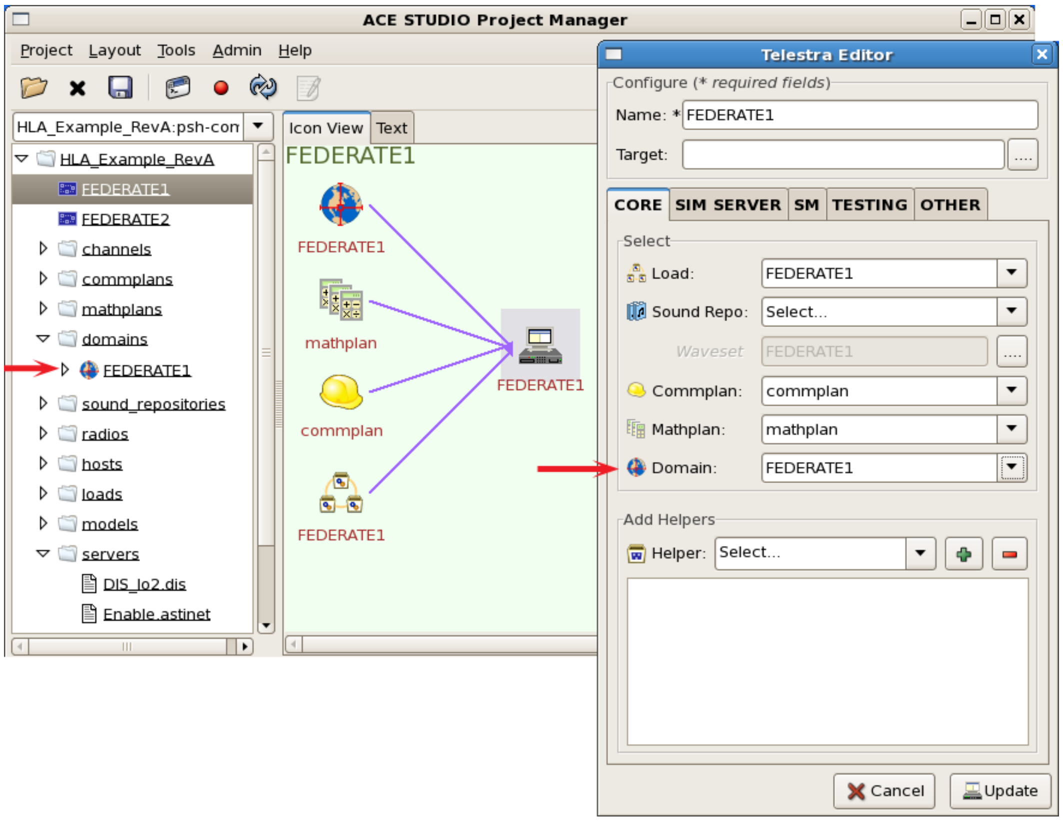
hf interface port

hf [✓ bcast] mcast

Step 5: Configure the Target to use the Domain and DIS Gateway

In this step you will set the Target to use the Gateway configuration file you just created.

1. In ACE Studio, navigate to the layout you wish to use, right-click on the Telestra icon, and select '**Edit**.'
2. Under the 'Core' tab add the Domain Helper name that contains the HLA Domain created in Step 3.



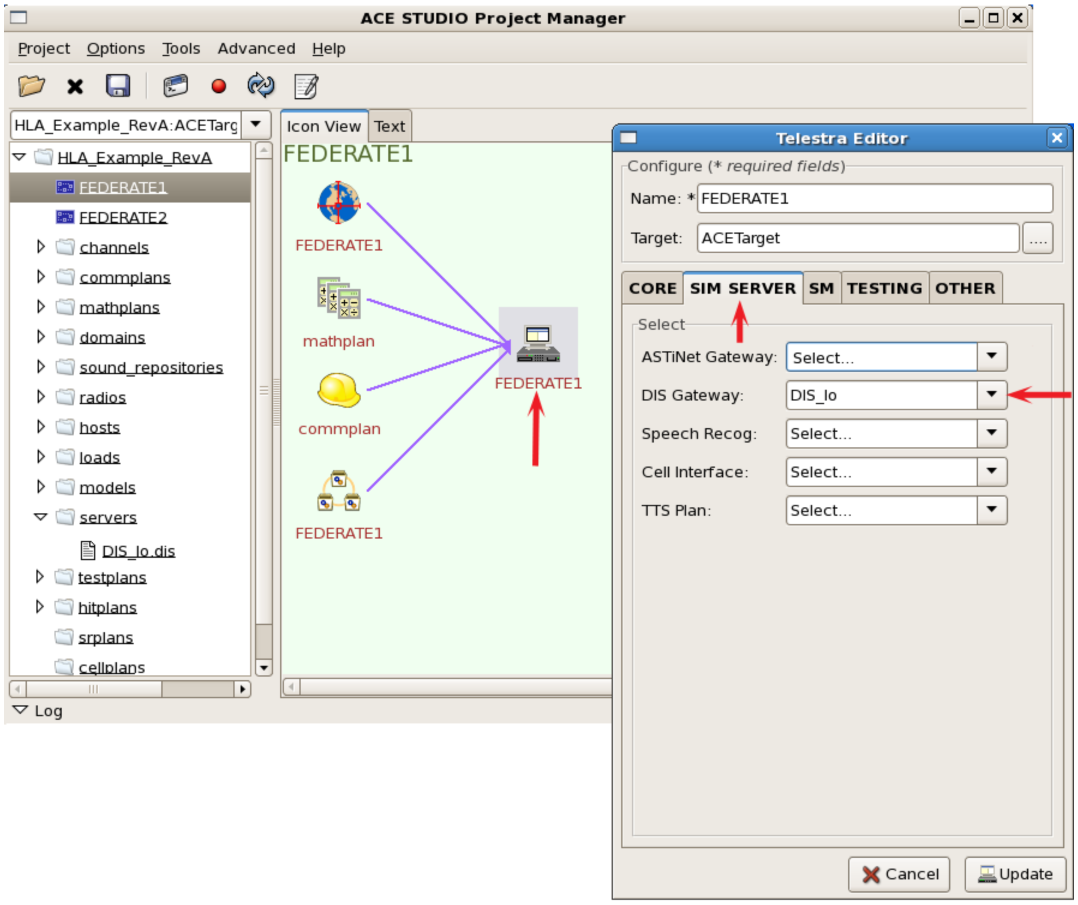
Configure (* required fields)
Name: * FEDERATE1
Target:

Select
Load: FEDERATE1
Sound Repo: Select...
Waveset FEDERATE1
Commplan: commplan
Mathplan: mathplan
Domain: FEDERATE1

Helper: Select...

3. Then navigate to the 'Sim Server' tab.
4. Select the DIS gateway configuration file that you just created. Then select '**Update**.'

   The Target is now configured to output network traffic for HLA.



Configure (* required fields)

Name: * FEDERATE1

Target: ACETarget

Select

ASTiNet Gateway: Select...

DIS Gateway: DIS_lo

Speech Recog: Select...

Cell Interface: Select...

TTS Plan: Select...

5. Save the project before continuing with the next step.

Step 6: Configure the Target Radios for HLA

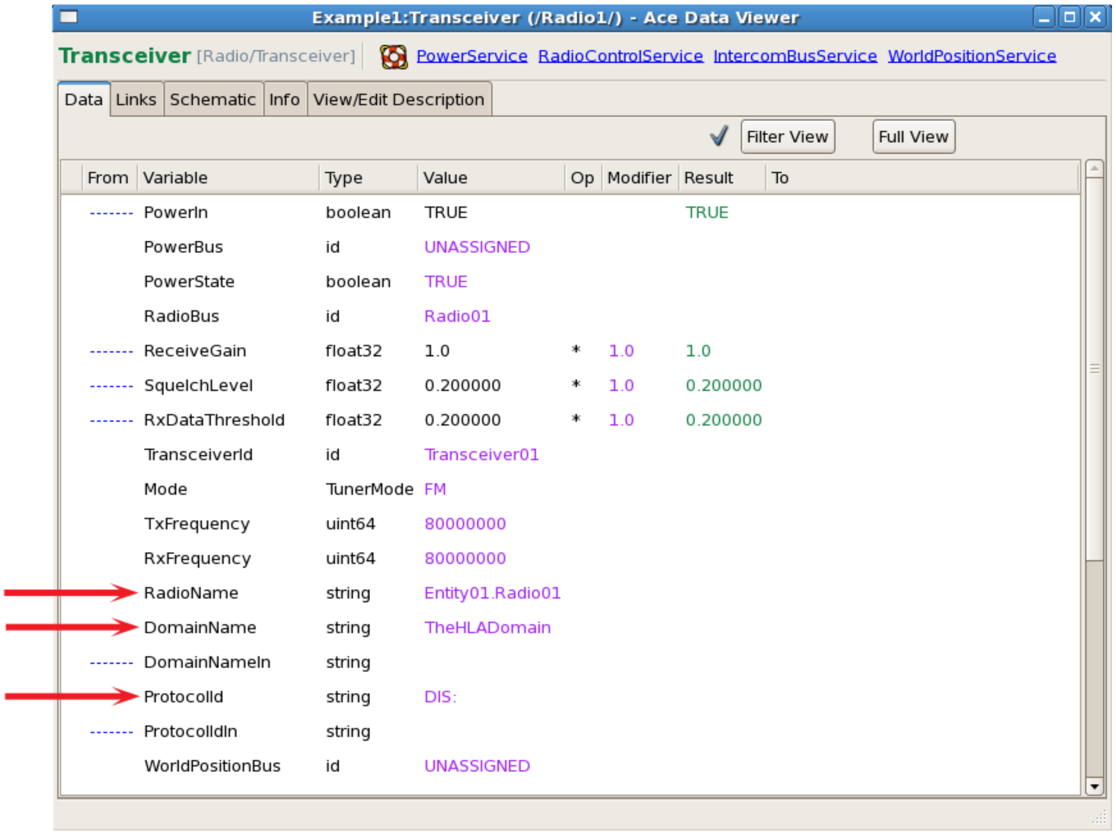
Note: The following section assumes the Target has a model with radios. See the ACE Studio Components Reference Guide (DOC-01-TELAS-CRG-4) for more information about radios.

1. Now that the Target is configured to output HLA traffic, select which radios are set up using HLA.
2. Navigate to the Model canvas and define the Domain, Protocol ID and RadioName in the Transceiver component.

   The Radio Name defines the HLA Radio name used in the HLA world.

   By selecting the proper Domain Name, the model is set to run in HLA. This must match the Domain that was created in the Domain Helper. “HLA” is used in this example.

   The Protocol ID sets the HLA Stream tag via the use of the DIS: syntax command. For example, DIS:1.2.3.4. sets the corresponding HLA Stream tag. If no specific Stream tag is required, set this to “DIS:”, which will make the radio unique.



From	Variable	Type	Value	Op	Modifier	Result	To
-------	PowerIn	boolean	TRUE			TRUE	
	PowerBus	id	UNASSIGNED				
	PowerState	boolean	TRUE				
	RadioBus	id	Radio01				
-------	ReceiveGain	float32	1.0	*	1.0	1.0	
-------	SquelchLevel	float32	0.200000	*	1.0	0.200000	
-------	RxDataThreshold	float32	0.200000	*	1.0	0.200000	
	TransceiverId	id	Transceiver01				
	Mode	TunerMode	FM				
	TxFrequency	uint64	80000000				
	RxFrequency	uint64	80000000				
	RadioName	string	Entity01.Radio01				
	DomainName	string	TheHLADomain				
-------	DomainNameIn	string					
	ProtocolId	string	DIS:				
-------	ProtocolIdIn	string					
	WorldPositionBus	id	UNASSIGNED				

To continue the HLA configuration see section ‘**8.7. Advanced Parameters**.’

8.6. Two Platform Solution

In the two platform solution, one of the Target platforms is configured solely as the communications and aural cue system (Communications Target) while the other Target is configured as the HLA Gateway (HLA Target).

8.6.1. Configuring the Communications Target

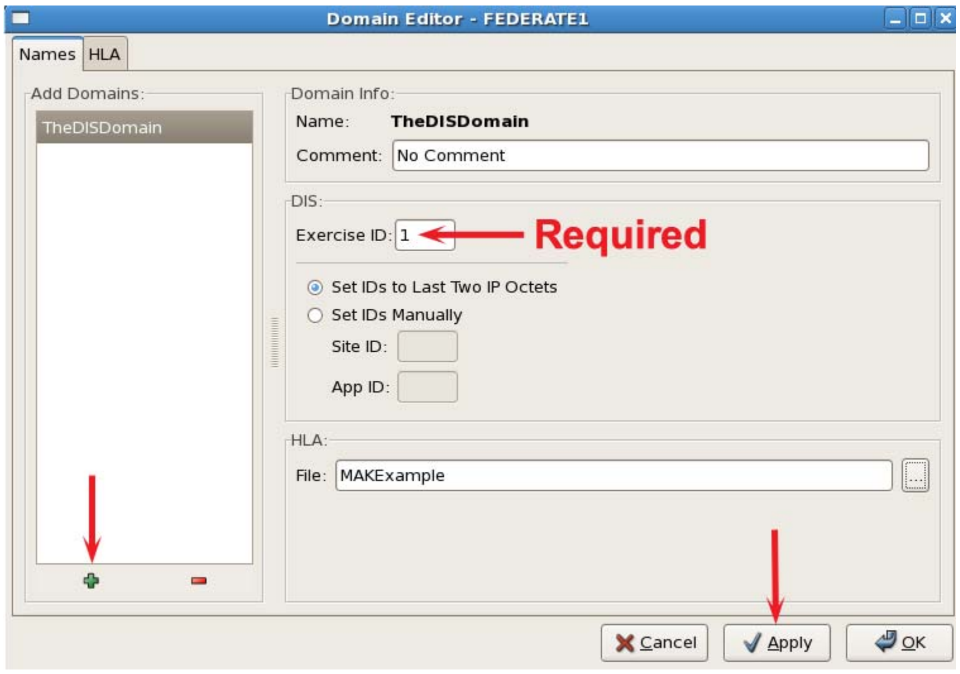
For this example, the communications Target will be configured to talk to the HLA Target. The domain will send DIS-like traffic to the HLA Gateway.

Step 1: Add a New Domain

Domain Icon

1. In ACE Studio, navigate to the Domain section using the left menu bar and then double-click the icon.
2. Select the green '+' button to add a new domain.
3. Enter a name for the domain, such as DIS as shown in the example below.
4. Under DIS, enter the exercise ID and define the Site and Application IDs. **The exercise ID must be set to 1**.

   The Site and Application IDs must be defined because they set the first two parts of the 64 bit stream tag identifier in the HLA world.
5. Press '**Apply**' to submit the configuration.


Domain Editor - FEDERATE1

Names | HLA

Add Domains:
- TheDISDomain

Domain Info:
Name: **TheDISDomain**
Comment: No Comment

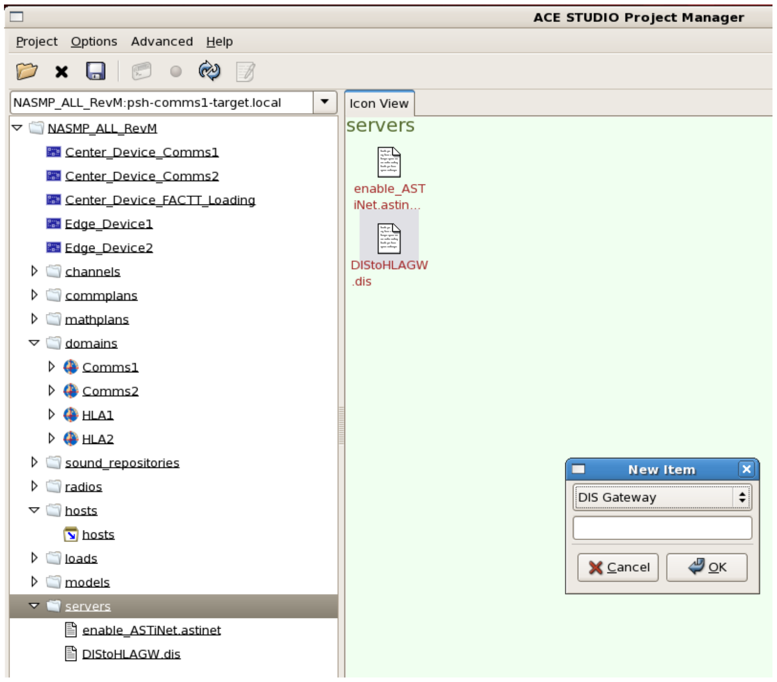
DIS:
Exercise ID: 1

[✓ Set IDs to Last Two IP Octets]
Set IDs Manually
Site ID:
App ID:

HLA:
File: MAKExample

Step 2: Add a DIS Gateway

1. In ACE Studio, navigate to the servers section and right-click on the canvas to add a DIS Gateway.



2. Enter a name such as ‘DIStoHLAGW’ and press ‘**OK**’.
3. Open the DIS Gateway you just created by double-clicking on it. Enter the following:

   Version = 132

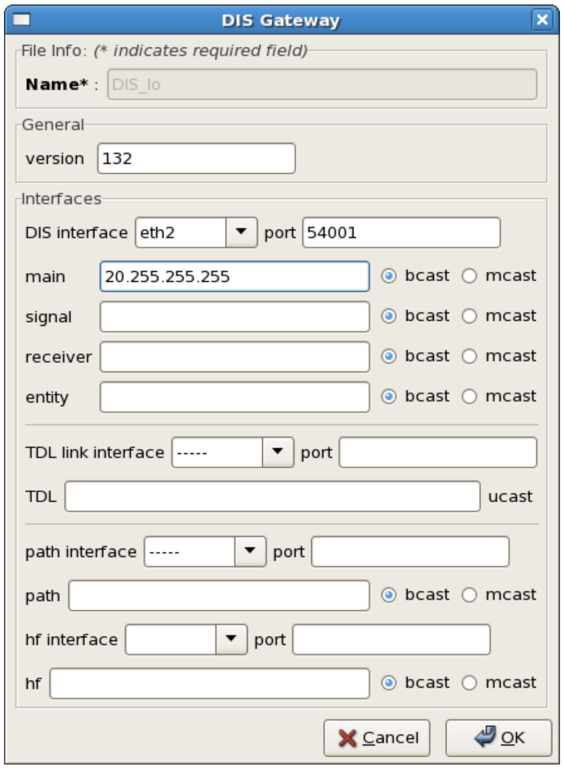
   > **Note**: Setting the version to 132 is required in order for the HLA Target to properly process the packets.

   Interface = eth2

   Port = 54001

   Main = Subnet Broadcast of eth2 and bcast is selected. The default Telestra *eth2* broadcast IP is 20.255.255.255.
4. Leave the remaining fields blank and press ‘**OK**’.

Note that these settings **must** match the ASTi HLA Network Settings defined in the '**Configuring the HLA Target**' section.



Name* : DIS_lo

General

version 132

Interfaces

DIS interface eth2 port 54001

main 20.255.255.255 [✓ bcast] mcast

signal [✓ bcast] mcast

receiver [✓ bcast] mcast

entity [✓ bcast] mcast

TDL link interface ----- port

TDL ucast

path interface ----- port

path [✓ bcast] mcast

hf interface port

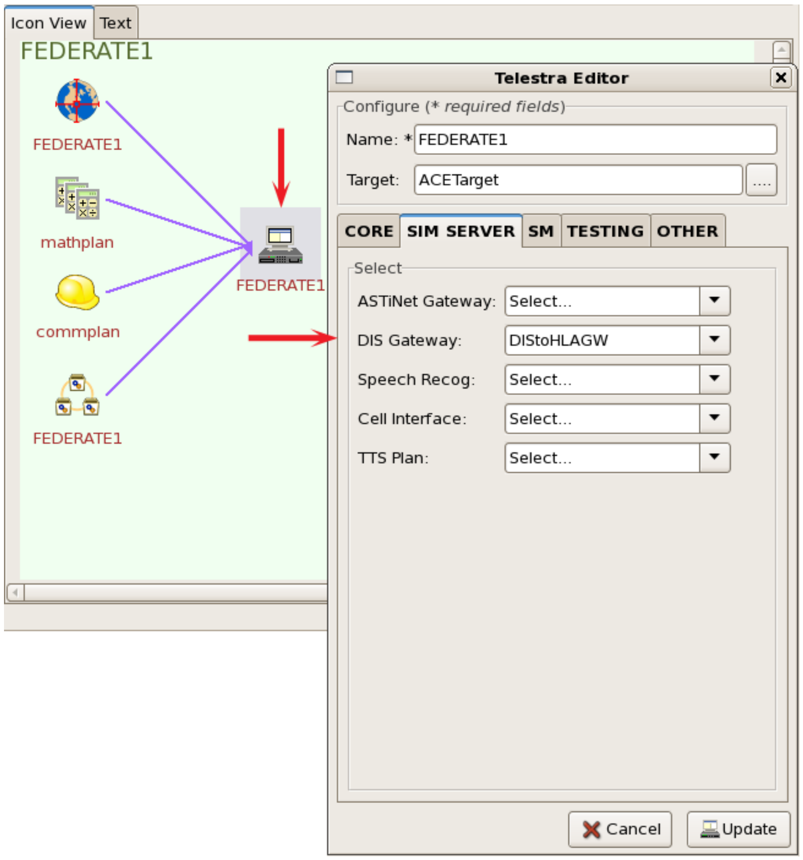
hf [✓ bcast] mcast

Step 3: Configure the Target to use the DIS Gateway

In this step you will set the Target to use the Gateway configuration file you just created.

1. In ACE Studio, navigate to the layout you wish to use, right-click on the Telestra icon, and select ‘**Edit**.’
2. Under the ‘Core’ tab add the Domain Helper name that contains the Domain created in Step 1.
3. Then navigate to the ‘Sim Server’ tab.
4. Select the DIS gateway configuration file that you just created. Then select ‘**Update**.’

   The communications Target is now configured to output DIS network traffic to the HLA Target. The DIS traffic will be translated into HLA traffic.



Configure (* required fields)

Name: * FEDERATE1

Target: ACETarget

Select

ASTiNet Gateway: Select...

DIS Gateway: DIStoHLAGW

Speech Recog: Select...

Cell Interface: Select...

TTS Plan: Select...

5. Save the project before continuing with the next step.

Step 4: Configure the Target Radios for DIS

Note: The following section assumes the Target has a load with radios and entities. See the ACE Studio Components Reference Guide (DOC-01-TELAS-CRG-4) for more information about radios, RCUs and entities.

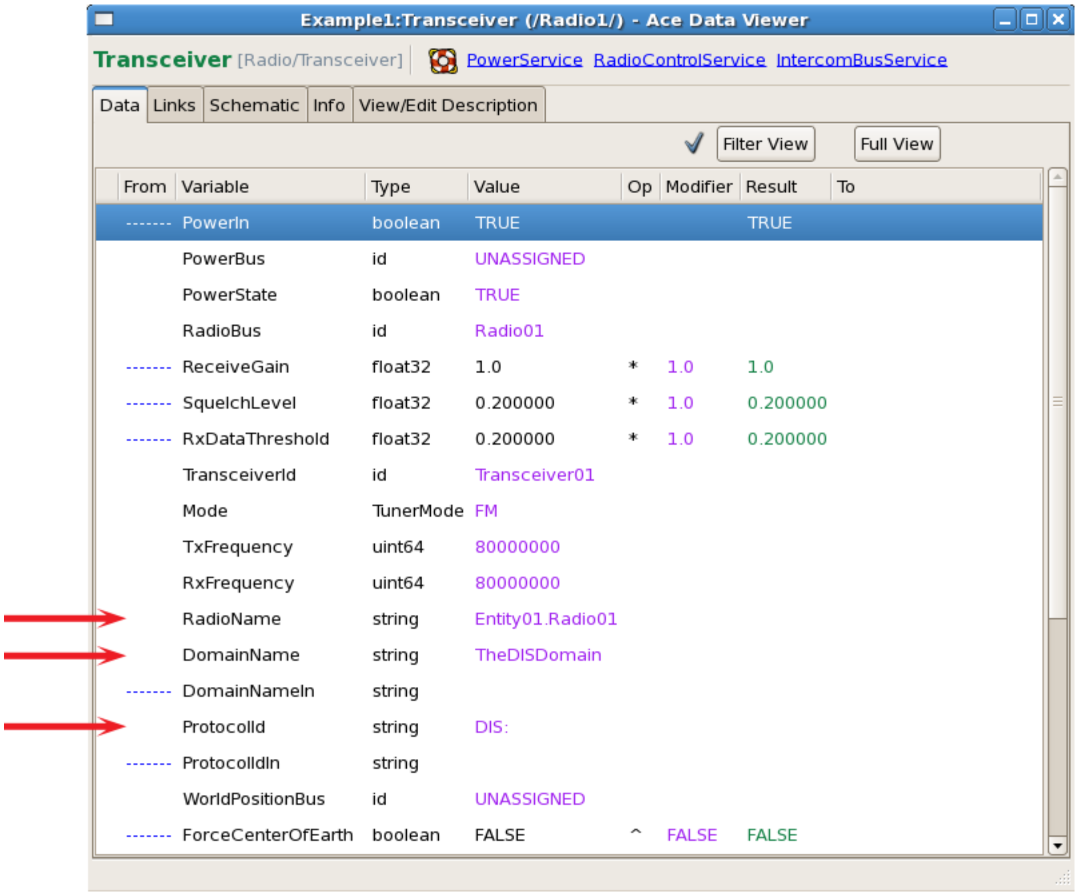
1. Now that the Target is configured to output DIS traffic, setup radios using DIS (later converted to HLA via HLA Target).

2. Navigate to the Model canvas and define the Domain, Protocol ID and RadioName in the Transceiver component.

   The **RadioName** defines the HLA Radio name used in the HLA world.

   By selecting the proper **Domain Name**, the model is set to run in HLA. This must match the Domain that was created in the Domain Helper, 'TheDISDomain' is used in this example.

   The **Protocol ID** sets the HLA Stream tag via the use of the DIS: syntax command. For example, DIS:1.2.3.4. sets the corresponding HLA Stream tag. If no specific Stream tag required set to "DIS:", which makes the radio unique automatically.

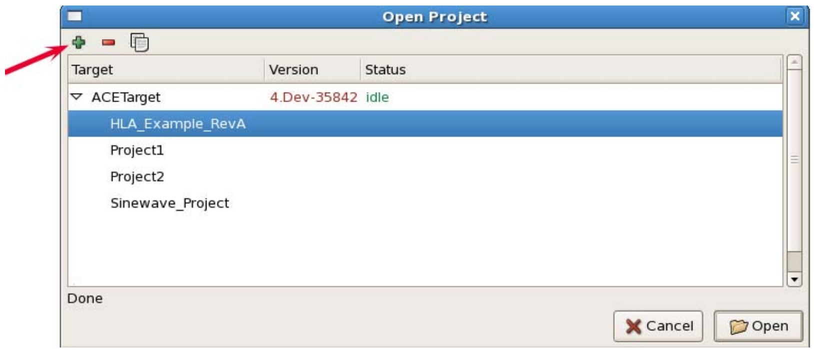


From	Variable	Type	Value	Op	Modifier	Result	To
-------	PowerIn	boolean	TRUE			TRUE	
	PowerBus	id	UNASSIGNED				
	PowerState	boolean	TRUE				
	RadioBus	id	Radio01				
-------	ReceiveGain	float32	1.0	*	1.0	1.0	
-------	SquelchLevel	float32	0.200000	*	1.0	0.200000	
-------	RxDataThreshold	float32	0.200000	*	1.0	0.200000	
	TransceiverId	id	Transceiver01				
	Mode	TunerMode	FM				
	TxFrequency	uint64	80000000				
	RxFrequency	uint64	80000000				
	RadioName	string	Entity01.Radio01				
	DomainName	string	TheDISDomain				
-------	DomainNameIn	string					
	ProtocolId	string	DIS:				
-------	ProtocolIdIn	string					
	WorldPositionBus	id	UNASSIGNED				
-------	ForceCenterOfEarth	boolean	FALSE	^	FALSE	FALSE	

3. Save the load and project before continuing to the next section.

8.6.2. Configuring the HLA Target

In ACE Studio, select the HLA Target from the Project menu before beginning this section. Add a new Project to the HLA Target. This project will contain the HLA configuration files.



Target	Version	Status
ACETarget	4.Dev-35842	idle
HLA_Example_RevA		
Project1		
Project2		
Sinewave_Project		

Step 1: Install and Activate the RTI

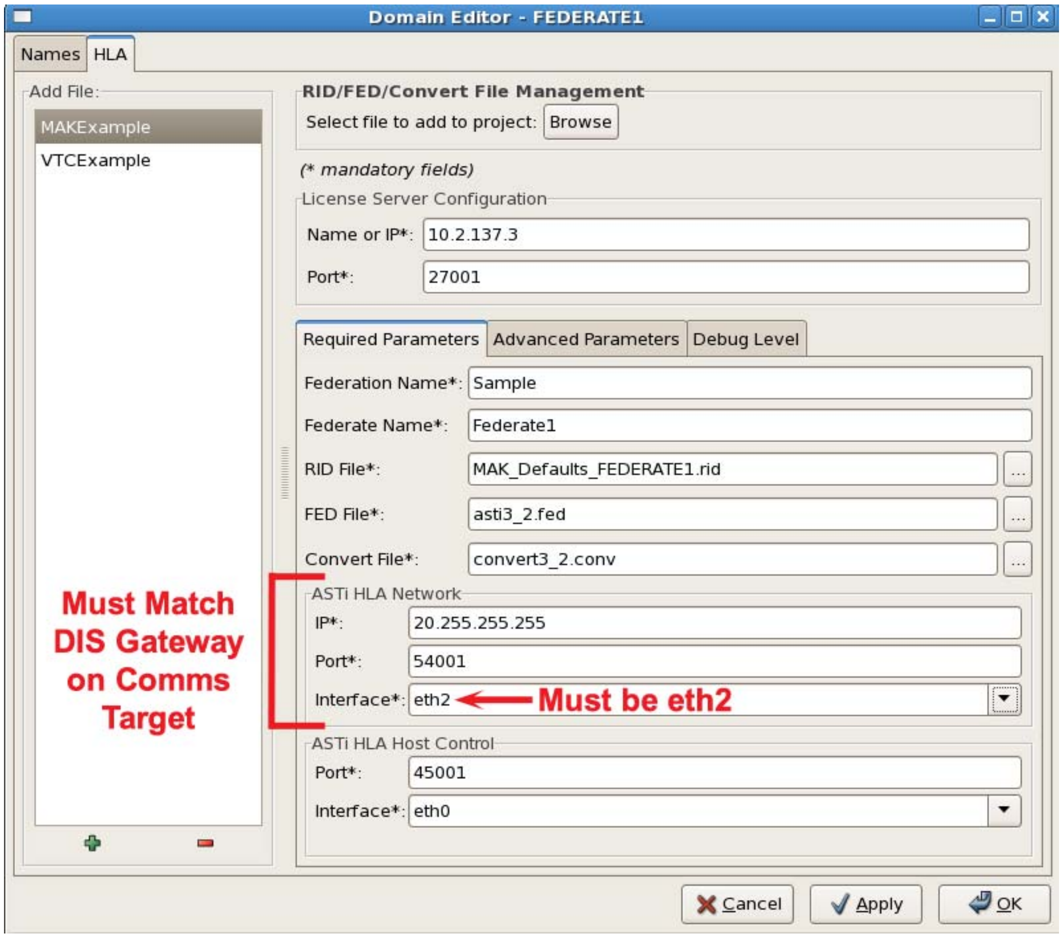
Use RMS to install and activate the RTI. Refer to the Telestra 4 Remote Management System 4 User Guide (DOC-01-TEL4-RMS4-UG-4) for details. You can find this and all ASTi documentation at www.asti-usa.com/support/document.

Step 2: Add a New HLA Configuration File

Domain Icon

The HLA Configuration file defines the Federation name, Federate Name, RID File, FED File, etc. The HLA configuration section consists of the required parameters, advanced parameters and debug level. Multiple Configuration files can be defined and used by the Target platform. This allows the user to create multiple HLA configurations and switch configurations, by simply changing the HLA configuration file.

To add a new HLA configuration file open your Domain, select the HLA tab, and select the '+' symbol. Below is an example of a standard MAK configuration files.



Select file to add to project:

(* mandatory fields)

License Server Configuration

Name or IP*: 10.2.137.3

Port*: 27001

Federation Name*: Sample

Federate Name*: Federate1

RID File*: MAK_Defaults_FEDERATE1.rid

FED File*: asti3_2.fed

Convert File*: convert3_2.conv

ASTi HLA Network

IP*: 20.255.255.255

Port*: 54001

Interface*: eth2

ASTi HLA Host Control

Port*: 45001

Interface*: eth0

Fill in the **Required Parameters** options. Parameter definitions are as follows:

License Server Configuration

The license server configuration defines the server IP address or hostname and defines the license server port.

Federation Name

The federation name is the name of the default exercise federation that the Target will be joining.

Federate Name

This is the name of the federate as seen by the federation. Each Target appears as a federate in the federation. The federate name will default to ASTi_Target_1, where Target is the Telestra's hostname. If this federate name is not desired, enter the required federate name.

RID File

Select the RID file required for the RTI to operate. Note that you must first upload an RID file.

FED File

Select the FED file required for the exercise. Note that you must first upload a FED file to the project by selecting the 'Browse' button from under RID/FED/Convert File management.

Convert File

Select the convert file required for the exercise. Note that you must first upload a Convert file to the project by selecting the 'Browse' button from under RID/FED/Convert File management.

ASTi HLA Network

This is the DIS-like network that the HLA Target uses to talk to the Communications Target. In this example, 20.255.255.255 and port 54001 were used. The values should match the DIS gateway created in Step 2 of section 8.6.1.

ASTi HLA Host Control

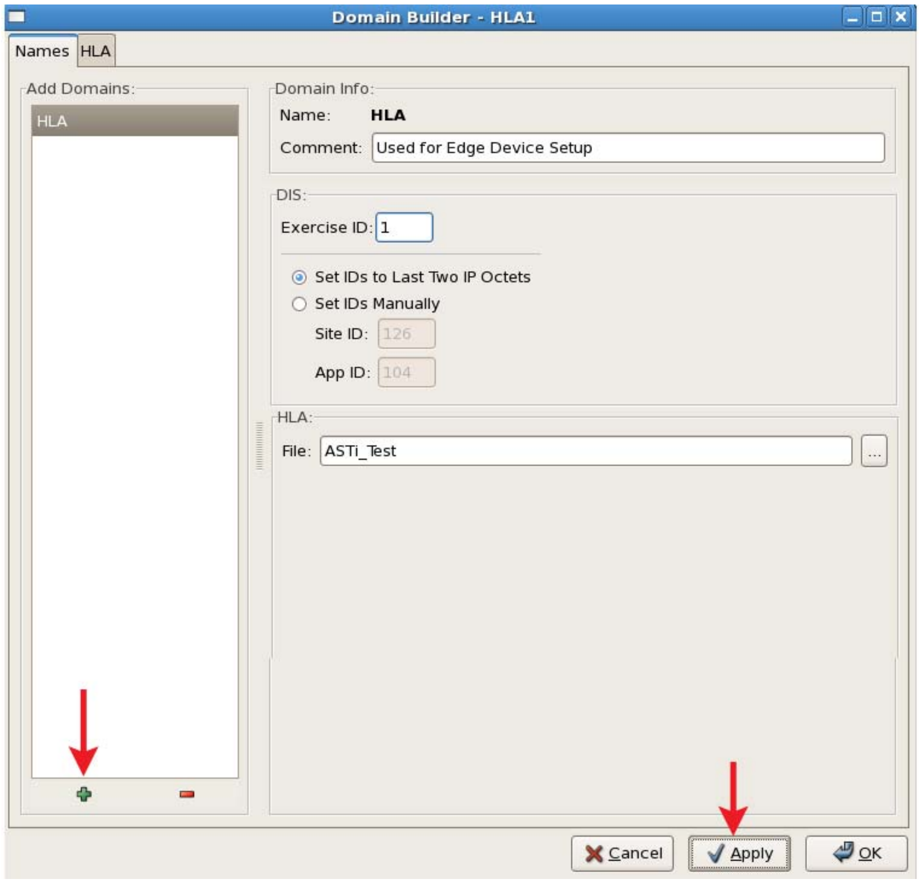
The ASTi HLA Host Control Interface defines which port and interface runs the TCP/IP host control interface. The details of using this interface are defined in the Telestra 4 Target Operation and Maintenance Manual (DOC-01-TEL4-TUG-1).

Port: Enter desired port number such as 54001.

Interface: Select the desired interface.

Step 3: Add a New Domain

1. In ACE Studio, navigate to the Domain section and double-click the icon.
2. Select the green '+' button to add a new domain.
3. Enter a name for the domain, such as 'HLA' as shown in the example below.
4. Set the Exercise ID to 1.
5. Under the 'HLA' add the **HLA Configuration file** that was created in Step 2.
6. Press 'Apply' to save the changes.

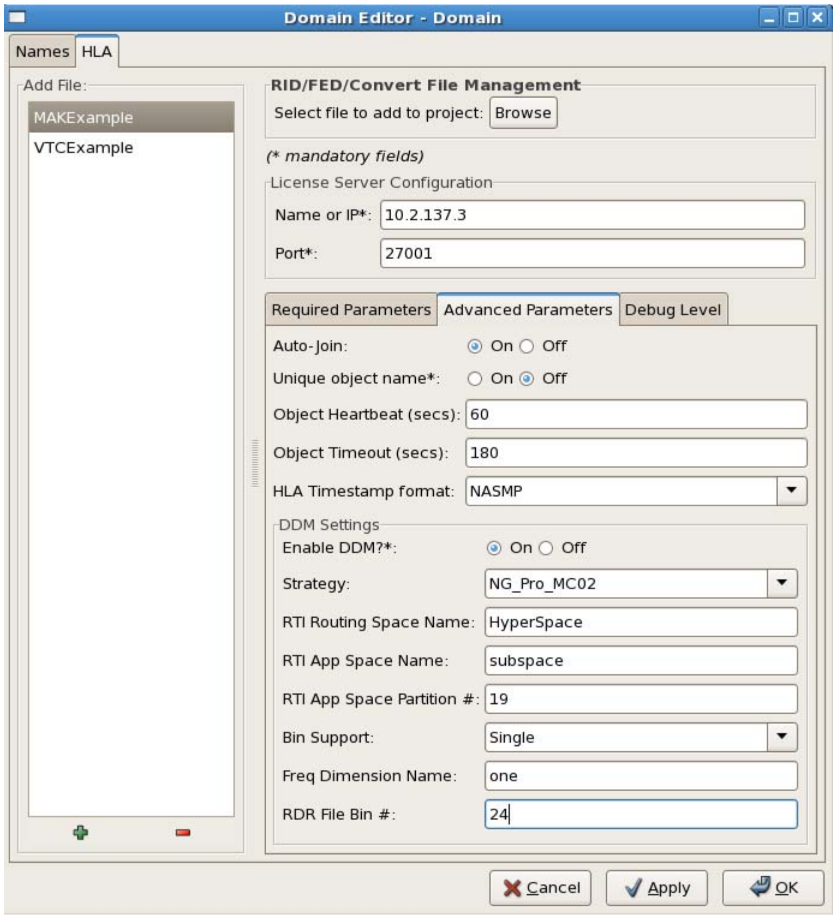


Name: HLA
Comment: Used for Edge Device Setup
Exercise ID: 1
[✓ Set IDs to Last Two IP Octets]
Set IDs Manually
Site ID: 126
App ID: 104
File: ASTi_Test

To continue the HLA configuration see section '**8.7. Advanced Parameters**.'

8.7. Advanced Parameters

Fill in the following **Advanced Parameters**.



Select file to add to project:

(* mandatory fields)

License Server Configuration

Name or IP*: 10.2.137.3

Port*: 27001

Auto-Join: [✓ On] Off

Unique object name*: On [✓ Off]

Object Heartbeat (secs): 60

Object Timeout (secs): 180

HLA Timestamp format: NASMP

DDM Settings

Enable DDM?*: [✓ On] Off

Strategy: NG_Pro_MC02

RTI Routing Space Name: HyperSpace

RTI App Space Name: subspace

RTI App Space Partition #: 19

Bin Support: Single

Freq Dimension Name: one

RDR File Bin #: 24

Auto Join

When this feature is enabled, the Telestra Federate will try to join the federation immediately after the layout has been installed. If using a default layout, the Telestra will attempt to join when the box boots up.

Unique Object Name

This is disabled by default. When enabled the federate name will be submitted to the radio object name. By default the radios are named as follows:

```
<Radio_Entity_Name>.<Radio_Name>.tx and .rx
```

When this feature is enabled the radio names will appear as:

```
<Federate_Name>.<Radio_Entity_Name>.<Radio_Name>.tx and .rx
```

This feature ensures that the network object names are unique. It also allows a user to load the same model on two or more platforms without having to make any naming modifications.

Object Heartbeat and Timeout Settings

Object Heartbeat and Object Timeout are disabled by default (i.e., they are set to 0). To enable these objects enter a value in seconds. If using the object heartbeat feature the user MUST define the timeout.

The Object Heartbeat defines the time in seconds that the radio objects will be updated. Traditionally in HLA object updates are only performed once and on a state change; however, with this feature enabled a user can cause the attribute updates to be sent out periodically. This is particularly useful when in connectionless mode. A common setting should be used across the network to avoid issues.

The Timeout settings are used in conjunction with the Object Heartbeat. This is the timeout period that will cause a federate to drop discovered remote Transmitter and Receiver objects. The value is defined in seconds. The recommended value should be two to three times the value defined in the Object Heartbeat setting.

HLA Timestamp format

Select the required Timestamp format. Current options include DMT and NASMP.

Enable DDM

Default = No.

DDM allows a federate to segregate and limit the flow of data to and from Targets using regions, i.e., publish with region, register object with region, send interaction with region, etc. There are many different DDM schemas such as Class Partitioning, Static Grid Partitioning, MC02 Strategy, etc. For more information on DDMs see section 2.4.4. and 2.4.5 in this document.

Strategy

Default = NG_Pro_MC02

Select the required DDM Strategy from the list:

NG_Pro_MC02 = NG Pro 4.0.X Millennium Challenge 2002

RTI Routing Space Name

Default = HyperSpace

Valid Entry = <ASCII TEXT>

This setting defines the RunTime Infrastructure Routing Space. This setting MUST match the current RID file and FED files to function correctly. See the Sample RID and FED files later in this section.

RTI App Space Name

Default = subspace

Valid Entry = <ASCII TEXT>

This setting defines the first dimension within the RTI Routing Space. Specifically, it defines the application space name. As discussed in a later section, this is broken into 200 partitions, only one of which will be used for radio communications. See the partition number below. This setting must match the current RID file and FED files to function correctly. See the Sample RID and FED files later in this section.

RTI App Space Partition #

Default = 23

Valid Range = 0 - 199

This setting defines the application space partition number that the Target will use in conjunction with the RTI. This defines the particular partition number for use with radio communications. This setting must match the current RID file to function correctly. See the Sample RID files later in this section.

Bin Support

Default = Single

This setting defines the number of supported bins within the RDR file. Currently, the only available option is single. If you look at the RID file you will see that dimension ‘one’ has 25 associated bins. You will select one of those bins later in the configuration.

Freq Dimension Name

Default = one

Valid Entry = <ASCII TEXT>

This setting defines the second dimension within the RTI Routing Space. Specifically, it defines a dimension of one. The definition of this dimension will vary depending on the particular subspace. For the Target, a dimension of one is mapping to the RF Spectrum. This setting MUST match the current RID file and FED files to function correctly. See the Sample RID and FED files later in this section.

RDR File Bin #

Default = 24

Valid Range = 0-24

This setting defines the bin to use within the frequency dimension (i.e. ‘one’). Currently, we only provide support for a single bin. This bin is fixed and currently defaults to 24; however, it is configurable in the range 0 - 24. This setting is not defined in the RID or FED files. To determine what bin to use, refer to the non-ASTi supplied RDR file.

Note: All the files that follow ARE NOT ASTi-provided files. The files were obtained either through the RTI vendor and/or the HLA Federation POC. Below is a snapshot of the relevant DDM sections of the RID, FED, and RDR files. Consult the RID, FED or RDR documentation for details on these files.

Sample RID File for use with DDM NG_Pro_MC02 Strategy

Note: Entire RID file IS NOT shown; only the relevant sub-sections are shown. Consult RTI Documentation for details.

```
(DataDistribution
  (StrategyToUse MC02)


(SpaceOptions
      (HyperSpace
        (DimensionOptions
          (subspace (NumPartitions 1))
          (one (NumPartitions 1))
          (two (NumPartitions 1))


(NumberOfSubspacePartitions 200


    (default 1)


        (0  (ap_space1            (dimension1   1)    (dimension2   1)))
…………


        (23  (radio_space            (dimension1   25)   (dimension2   1)))
```

Sample FED File for use with DDM NG_Pro_MC02 Strategy

Note: Entire FED file IS NOT shown; only relevant subsections are shown. Consult Federation Agreement Documentation for details.

```
(FED
(Federation Federation123)
(FEDversion v1.3)
 (spaces
 (space HyperSpace
    (dimension subspace)
    (dimension one)
    (dimension two)

(class CommunicationSystem
 (class RadioReceiver
     (attribute HostObjectIdentifier best_effort receive HyperSpace)
………

 (class RadioTransmitter
     (attribute AntennaPatternData best_effort receive HyperSpace)
     (attribute Encryption best_effort receive HyperSpace)
…………

(class RadioTransmission best_effort receive HyperSpace
     (parameter StreamTag)
………
```

Sample RDR File for use with DDM NG_Pro_MC02 Strategy

Note: Entire RDR file IS NOT shown; only relevant sub-sections are shown. Consult RTI/ RDR Documentation for details.

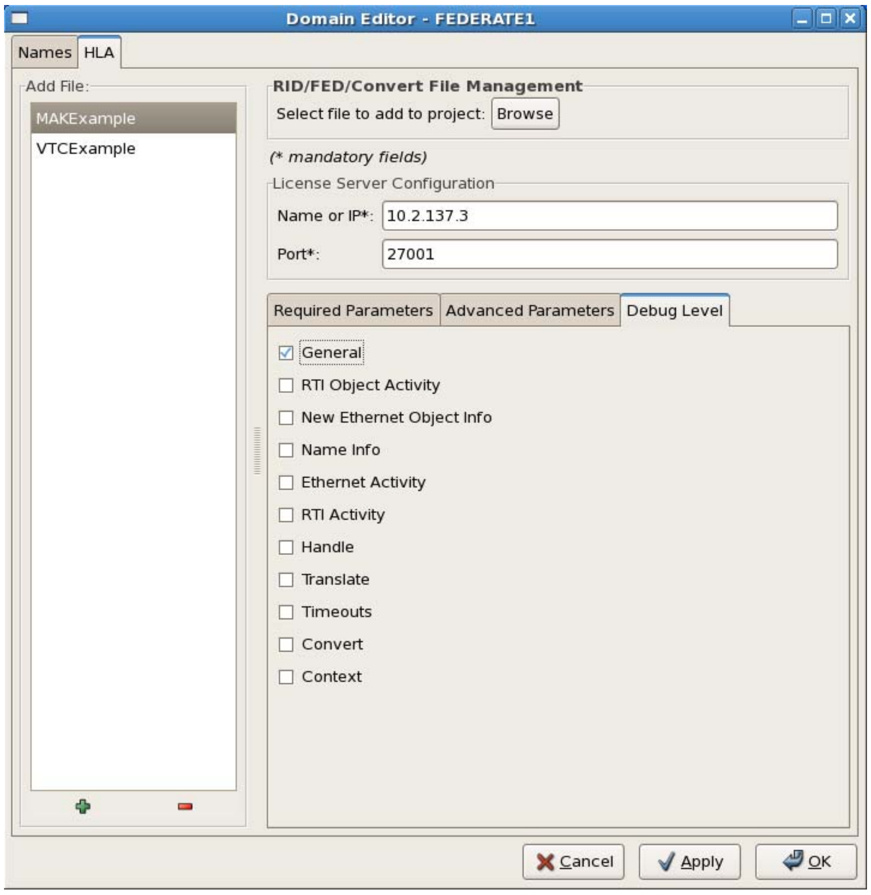
RDR File

```
"RADIO_DIMENSIONS" {
   "partition"        ;; use the partitioned scheme
   (scale 1000000.0)   ;; all values below are Mhz
   (values
      ;; 26 values divide the range into 25 bins, numbered 0 through 24...

        0.0    ;;     0hz  >= bin0  <   15Mhz    ;; IWEG/DCE
…………..
      6000.0    ;;    6Ghz  >= bin24 <    8Ghz    ;; ASTI Radio Communications
```

8.8. Debug Level

Select ‘On’ to turn on basic HLA logging.

For additional debug and logging capabilities select one of the ten (10) options available. Use caution when doing this, as selecting multiple debug options can cause a massive amount of logging to occur. In general, the log should be turned on when debugging HLA issues. Otherwise the recommended setting is OFF.



Name or IP*: 10.2.137.3

Port*: 27001

[✓] General
[ ] RTI Object Activity
[ ] New Ethernet Object Info
[ ] Name Info
[ ] Ethernet Activity
[ ] RTI Activity
[ ] Handle
[ ] Translate
[ ] Timeouts
[ ] Convert
[ ] Context

8.9. What is MC02 DDM?

The RTI Data Distribution Management services provide the framework for flexibility and scalability within a Federation. Facilitating a reasonable DDM scheme over a distributed training environment not only reduces the amount of network bandwidth used over federation network infrastructure, but also reduces the processing strain and memory usage required of each federate. This is because DDM uses a concept called partitioning to separate interest streams within the federation, such that each federate only subscribes to the things that are of interest. On the transmitting side, each federate only publishes to the proper interest streams. Therefore, any interest streams that a receiver or subscriber does not explicitly request do not get sent to that federate.

The concept of partitioning relies on the assertion that not every federate needs to know the state of the entire synthetic battle space. Federates are often only interested in unique subsets of the objects and interactions being published over the federation.

The RTI provides a second set of services for partitioning and filtering data, called Declaration Management (DM). Declaration Management uses class subscriptions to allow each federate to tell the RTI what they are interested in. Multicast filtering, which is used with DDM, cannot be used to support this approach. The RTI specification requires that all the attributes of a class always be published to the same space. This requirement becomes problematic when the FOM in question is hierarchical in nature (i.e. having a class structure where subclasses inherit attributes or parameters from parent classes). For this reason, and since the IEEE 1516 specification removed the concept of routing spaces altogether, one approach is to use DDM with a single routing space, called “HyperSpace”. This single routing space is segmented by using multiple dimensions (see definitions in next section), the first of which is used as an enumerated value that essentially breaks up the single routing space into mini-routing spaces, or “application spaces”. Each segment is then tied to a multicast group(s), which is used by each federate for publication and subscription to the interest stream associated with that segment. It should also be noted that this approach makes it easier to upgrade to 1516 RTI implementations in the future, since “no routing spaces” is essentially equivalent to one routing space.

An important note here is that ASTi did not develop the MC02 DDM Strategy. We have added software support for MC02 DDM Strategy. Therefore, the end user is expected to be familiar with the basic MC02 DDM concepts. Please refer to the appropriate federation document for more information. This is not provided by ASTi.

8.9.1. DDM Terms and Definitions

This section attempts to provide definitions for various terms often used when discussing DDM and related concepts.

Routing Space

A routing space is a collection of dimensions. These spaces are defined in the Federation Execution Data (FED) file, and provide the basis for DDM addressing. Each object attribute and interaction in the FED file is associated with one of these routing spaces, which are used for both publishing and subscribing. Note that only one routing space is used by the MC02 strategy. Also note that the terms “routing space” and “application space” (see below) can sometimes be easily confused. Remember that “routing space” usually refers to an RTI routing space in use (in our case, HyperSpace), while “application space” refers to a federation-supplied subspace.

Region

A region is also known as a subspace of a routing space, and is represented as a collection of extents. The region is the key to the DDM addressing concept and is used for both publishing and subscribing to object attribute updates and interactions.

Application Space

An extension of the RTI routing space concept, application spaces allow federation developers to create their own “internal” routing spaces, based on the needs of the applications within a federation. Within MC02 strategy, an application space can also be thought of as a subspace of an RTI routing space, and is associated with a list of object classes. MC02 straighter will only use one RTI routing space, therefore it is necessary to use one dimension in this space to create subspaces or application spaces.

Update Region

An update region should be specified for sending attribute updates and interactions across the federation. This is done through the service ‘`registerObjectInstanceWithRegion`’ to direct the RTI to send attribute updates through a region. In the same way, ‘`sendInteractionWithRegion`’ is used to direct the RTI to send an interaction through a region. It is useful to think of a region as an “interest stream”.

Subscription Region

Subscription regions are used in the same way as update regions. The RTI DDM services ‘`subscribeObjectClassAttributesWithRegion`’ and ‘`subscribeInteractionClassWithRegion`’ should be used when subscribing to object attributes and interactions, respectively. It is useful to think of a region as an “interest stream”.

8.10. HLA Troubleshooting and Debugging

If the join process fails, check the following:

- Check the RTI Known Issues list (Vendor specific)
- Verify the FED/Convert files. Revert to ASTi SOM and convert file for interim testing. For example, try loading the following default FED and Convert files:
  - asti3_2.fed
  - convert3_2.conv
- Verify the RTIEXEC is running. Do you see the federate attempting to join? Are there any error messages? Contact the RTI Vendor for other debug ideas.
- Verify the RID file settings are correct and match the RTIEXEC endpoints. Contact RTI vendor to verify RID file settings.
- Verify the license server is running. Can you ping license server hostname from the Target? Is DNS required/setup if you are using the hostname.
- Verify that the time on the Target and the time on the License Server are in sync. The time does not need to mach exactly.

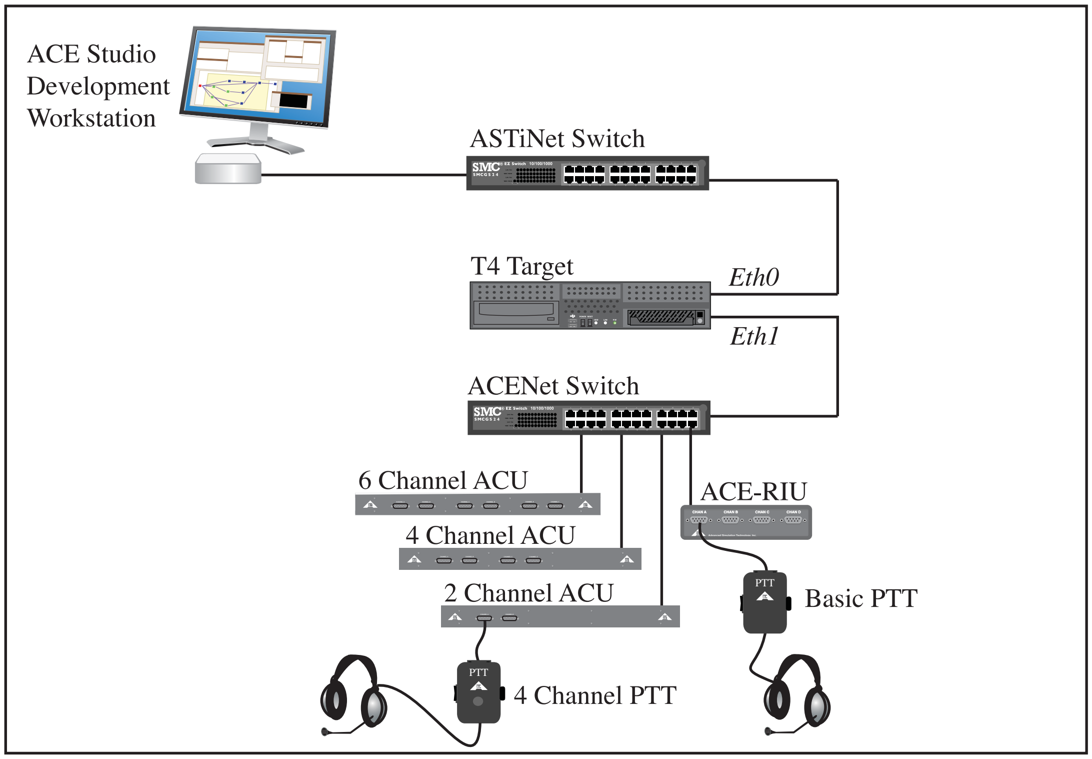
Examine the HLA Logs for guidance.

9.0. ACE Readiness Test

The ACE Readiness Test is designed to test the ACENet audio devices including ACUs and ACE-RIUs for audio inputs, outputs and PTT capability.

Before getting started, ensure that the ACE Studio Development Workstation and the T4 Target have the latest software version.

The Readiness Test hardware setup is shown in the system diagram below.

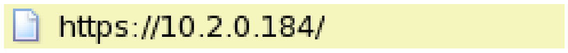


Step 1: Readiness Test Setup

1. Power on the Target and ACE Studio Development Workstation.
2. Log in to ACE Studio as

   `Username: aceuser   Password: aceuser`

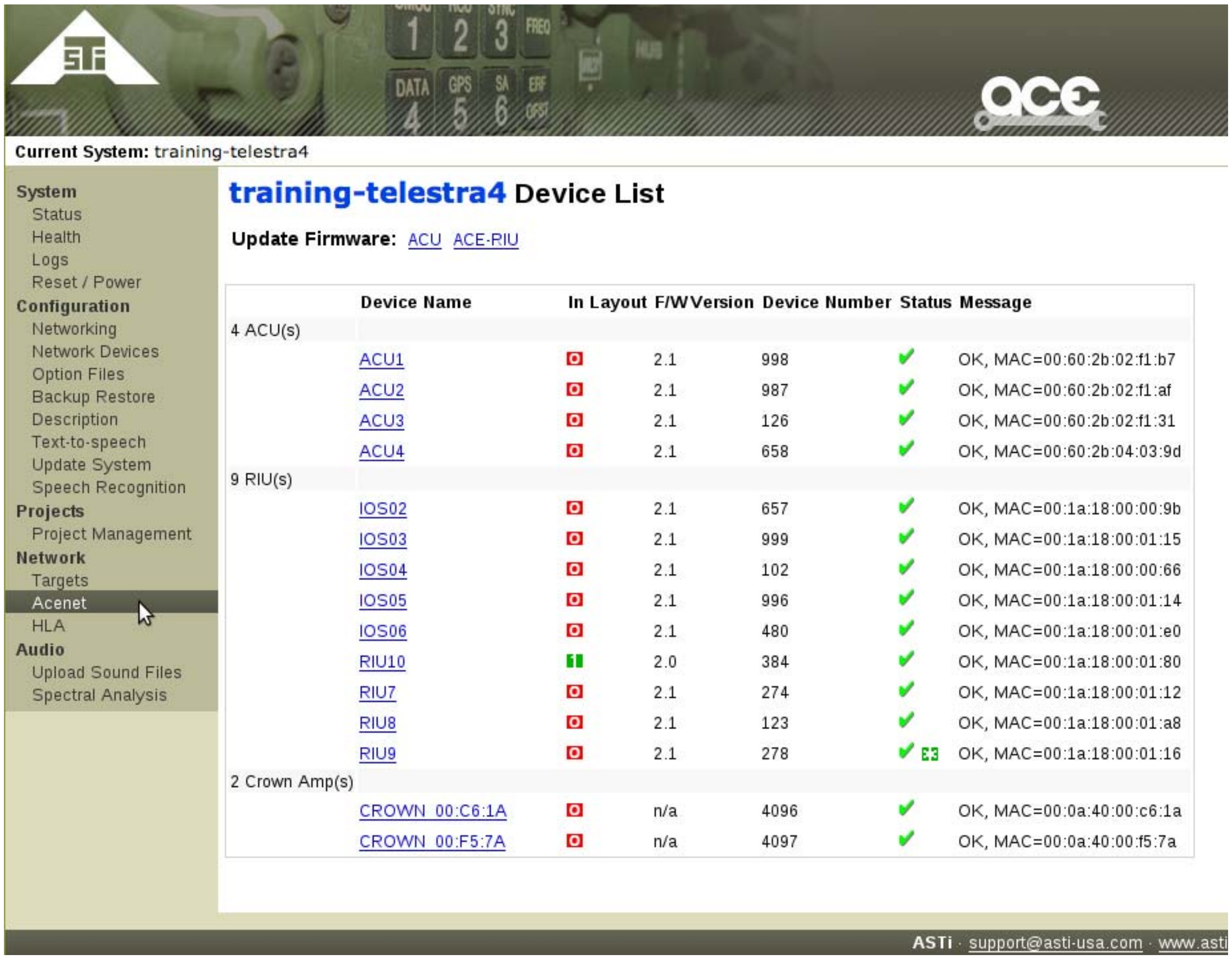
3. Double-click the web browser icon to open the web browser.
4. In the web browser's address bar, type the Target's IP address (Eth0) to open RMS.

   For example: 

   https://10.2.0.184/

5. In RMS, select the ACENet tab from the left menu bar.

   The RMS page for audio devices will list all devices recognized on the network. If you do not see all the hardware, reset and power on each device until it is recognized and displayed on the RMS page.



training-telestra4 Device List

Update Firmware: ACU ACE-RIU

	Device Name	In Layout	F/W Version	Device Number	Status	Message
4 ACU(s)						
	ACU1	🟥	2.1	998	✓	OK, MAC=00:60:2b:02:f1:b7
	ACU2	🟥	2.1	987	✓	OK, MAC=00:60:2b:02:f1:af
	ACU3	🟥	2.1	126	✓	OK, MAC=00:60:2b:02:f1:31
	ACU4	🟥	2.1	658	✓	OK, MAC=00:60:2b:04:03:9d
9 RIU(s)						
	IOS02	🟥	2.1	657	✓	OK, MAC=00:1a:18:00:00:9b
	IOS03	🟥	2.1	999	✓	OK, MAC=00:1a:18:00:01:15
	IOS04	🟥	2.1	102	✓	OK, MAC=00:1a:18:00:00:66
	IOS05	🟥	2.1	996	✓	OK, MAC=00:1a:18:00:01:14
	IOS06	🟥	2.1	480	✓	OK, MAC=00:1a:18:00:01:e0
	RIU10	🟩	2.0	384	✓	OK, MAC=00:1a:18:00:01:80
	RIU7	🟥	2.1	274	✓	OK, MAC=00:1a:18:00:01:12
	RIU8	🟥	2.1	123	✓	OK, MAC=00:1a:18:00:01:a8
	RIU9	🟥	2.1	278	✓ 🟩	OK, MAC=00:1a:18:00:01:16
2 Crown Amp(s)						
	CROWN_00:C6:1A	🟥	n/a	4096	✓	OK, MAC=00:0a:40:00:c6:1a
	CROWN_00:F5:7A	🟥	n/a	4097	✓	OK, MAC=00:0a:40:00:f5:7a

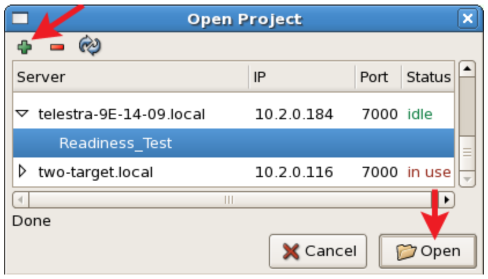
6. Ensure that the audio devices have the latest firmware. If they do not, install the latest firmware on all the devices.

   For the ACU, see the ACENet Communication Unit (ACU) Technical User Guide (DOC-01-TEL4-ACU-UG-1) section '**Updating Firmware**' for detailed instructions.

   For the ACE-RIU, see the ACE-RIU Technical User Guide (DOC-01-TEL4-AR-UG-1) section '**Updating Firmware**' for detailed instructions.

7. Open the ACE Studio software on the Development Workstation by navigating to the top left tool bar selecting **Applications** > **ASTi > ACE Studio**. This opens the ACE Studio Project Manager.

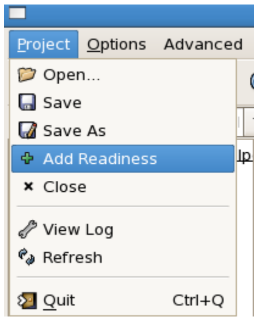
8. Select **Project** > **Open** and select the Target name. Select the '+' to open a new project on the Target. Name the new project (example Readiness_Test). **Note**: Do not use spaces in the project name.



Server	IP	Port	Status
telestra-9E-14-09.local	10.2.0.184	7000	idle
Readiness_Test			
two-target.local	10.2.0.116	7000	in use

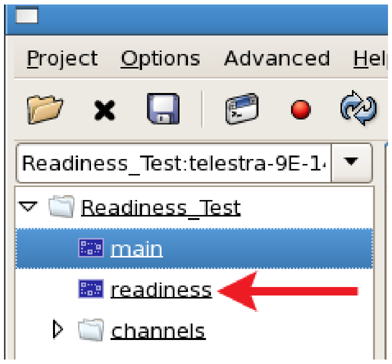
9. Click '**Open**' to access the Project Manager.

10. In the '**Project**' drop-down menu, select '**+ Add Readiness**' and click '**Yes**.'

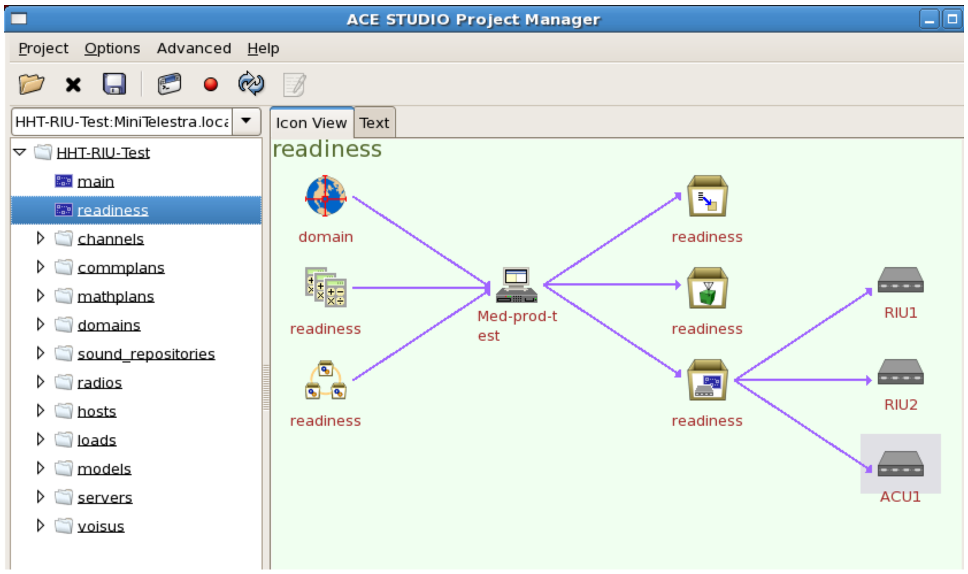


This will create the readiness layout, which is a readiness model.

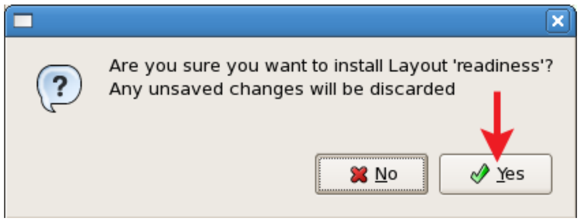
11. Under the Readiness_Test folder select '**readiness**' under Main to open the layout.



12. The readiness layout will display all of the connected audio devices on the network.

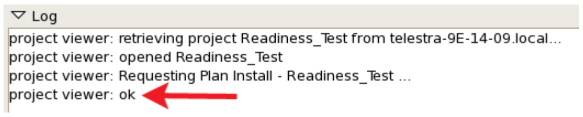


13. Select the '**Install Layout**' icon (top left), and when prompted click '**Yes**' to install the layout.



Are you sure you want to install Layout 'readiness'?
Any unsaved changes will be discarded

In the Log at the bottom of the screen, the Project Viewer will display '**OK**' which means the model is up and running.



```
project viewer: retrieving project Readiness_Test from telestra-9E-14-09.local...
project viewer: opened Readiness_Test
project viewer: Requesting Plan Install - Readiness_Test ...
project viewer: ok
```

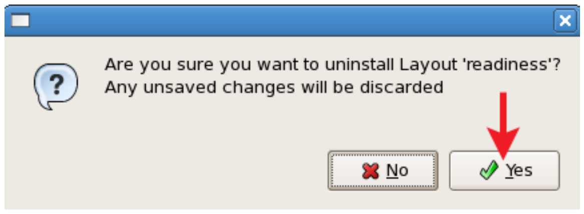
Step 2: Testing Audio

Test the audio devices for audio input, output, and PTT capability:

1. Press the PTT switch and speak into the headset microphone. You should hear yourself. If not, check the volume knob on the PTT.
2. If testing an ACU, check channels 1-4 on the PTT. The tones should change for each channel. **Note**: The ACE-RIUs do not use the 4 Channel PTT.
3. Check each channel on the device for audio input and output. For ACE-RIUs check channels A-D, and for ACUs check channels A-F (depending on hardware).

   All channels will play the same tone through the headset for audio output.

   The audio device is fine if all channels have audio input, output, and PTT capability.
4. Test the remaining audio devices in the Layout.
5. When complete, click the ‘**Uninstall**’ button and then click ‘**Yes**’.


Are you sure you want to uninstall Layout 'readiness'?
Any unsaved changes will be discarded

6. Select **Project** > **Close**.
7. Close the Project Manager window and click ‘**Yes**’ to quit ACE Studio.

10.0. Security

The Security package for ACE Studio is available for ACE 4.26 software version and later. See the Target Operation and Maintenance Manual (DOC-01-TEL4-TUG-1) for the ACE security package overview.

To install the Telestra 4 Security Software package see the ACE Studio Cold Start Procedure (DOC-02-TEL4-ASCS-1) section 5.0 “Installing Telestra 4 Security Software Package” for step-by-step instructions.

For additional information about Telestra 4 security, please see the ASTi website at:

```
http://www.asti-usa.com/support/faq/t4security.html
```

11.0. Warranty Information

The equipment is under warranty for a period of one (1) year following purchase. In the case of equipment upgrades, the warranty applies to the original date of shipment of individual components.

Other commercial equipment purchased or provided such as monitors, amplifiers, speakers, and fiber optic links are also covered under the one year warranty unless otherwise stated.

The warranty does not cover improper equipment handling or improperly packaged returns. Extended warranties are available. Contact ASTi for details: (703) 471-2104.

11.1. Repairs and Returns

If it becomes necessary to return equipment to ASTi please observe the following instructions:

1. Request an RMA number through the form on the ASTi web site: www://www.asti-usa.com/support/

   The receiving department at ASTi will not receive a repair without an RMA number.

2. When packaging the equipment in question, make sure it is well protected. ALWAYS DOUBLE BOX the DACS/Telestra. The inner container should employ some semi-rigid, contour-fitting foam, while the exterior container should use a more pliant, shock-absorbing material such as styrofoam peanuts. The device should be properly enclosed in an antistatic bag to prevent possible ESD damage. Failure to properly package the equipment during shipping could void the warranty.

3. Do not send accessory pieces such as rack mount kits, power supplies or software. Only include items that do not work.

4. The shipping label must include the RMA number.

5. Include a description of the problem including the serial number for the unit in question. Include point of contact information including name, telephone number, and equipment return address. Failure to include this information could extensively delay the return of the equipment.

6. Evaluation of equipment is performed free of charge. No work will be done without prior customer approval.

7. Customer is responsible for shipping charges to ASTi for warranty and non-warranty repairs.

8. Note that if equipment is not under warranty, a purchase order will be required to cover any repairs. ASTi will provide a quote for all non-warranty items, including return shipping. Customer is responsible for return shipping charges on non-warranty equipment.

9. Equipment still under warranty will be shipped back via Federal Express, unless otherwise directed. ASTi is responsible for return shipping charges on domestic items under warranty.

10. If equipment is not received by ASTi within thirty (30) days of the RMA number issuing date, the request data and number issued will be closed and designated as unused.
11. Any items received from customers without RMA numbers or appropriate contact information included with shipment will not be tested. After sixty (60) days, ASTi reserves the right to scrap all hardware received in this condition.
12. **International customers** must include the correct product value on all shipping documents. Contact ASTi for proper harmonized tariff codes. The customer is responsible for duties, taxes and fees incurred in shipment of the equipment.

11.2. Disclaimer and Warnings

There are NO user-serviceable components in this device. Opening the chassis will void the warranty.

Appendix A: Software Version Release Notes

Please see the ASTi website for all software release notes.

```
www.asti-usa.com/support/faq/telestra4/12.html
```

Appendix B: Technical Specifications

The following technical specifications are for the small footprint system (part number T4-DEV-01).

Specification	Suggested Range
Line Voltage	100 – 240 V AC
Frequency	50Hz – 60Hz, single phase
Maximum Continuous Power	110W
Operating Temperature	50° to 95° F (10° to 35° C)
Storage Temperature	-40° to 116° F (-40° to 47° C)
Relative Humidity	5% – 95% non condensing
Maximum Altitude	10,000 feet

Size and Weight	
Height	2 inches (5.08 cm)
Width	6.5 inches (16.5 1cm)
Depth	6.5 inches (16.51 cm)
Weight	2.9 pounds (1.31 kg)

The following technical specifications are for the Telestra 4 platform.

Target	Degrees Fahrenheit	Degrees Celsius	Humidity	Altitude
Operation Environment	50 – 90 F	10 – 32.2 C	20% – 95%	0 – 8,000 ft. (2,438 meters)
Storage Environment	32 – 135 F	-10 – 55 C	10% – 95%	0 – 8,000 ft. (2,438 meters)

The Telestra 4 platform is a standard 2U, 19” rackmount system.

Telestra 4	Depth	Width	Height
Standard 2U	22 3/4”	17”	3 1/2”
With Front Rackmount	24”	19” (with thumb screws)	3 1/2”
With Middle Rackmount	22 3/4”	19”	3 1/2”