Verastream Host Integrator

Design Tool Guide

Table of contents

1. Welcome to Host Integrator

Verastream integration encapsulates host functionality into services for rapid re-use in new applications. Verastream Host Integrator (VHI) contains these components:

- Design Tool

  You use the Design Tool to build host application models quickly and easily by navigating through legacy host applications and selecting the appropriate fields.

- Web Builder

  After you create a model, you use Web Builder to generate and deploy a variety of Web projects; including HTML 5 Web applications and objects for both Java and .NET environments. Web Builder generates all the necessary project files so that the source code can be quickly viewed, modified, and rebuilt using industry standard tools such as the latest versions of Visual Studio, Eclipse, and Visual Basic.

- Host Integrator Server

  The Host Integrator server provides an enterprise production environment in which to deploy your models. The Host Integrator servers support failover and load balancing using the Administrative Console as a central management hub.

- Administrative Console

  You use the Administrative Console to deploy, configure, monitor, and optimize your Host Integrator models and servers in a production environment

Take the tutorial Spend 30 minutes and use Design Tool and Web Builder to build and deploy a simple Host Integrator application.

1.1 Terminal Sessions from the Start Menu

You can install VHI and have immediate access to your host. From the Start menu, choose either 3270 Terminal Session or 5250 Terminal Session, depending on your host type. You can also enter a known address such as `http://<server-name>:8081/Terminal3270` into your browser. An HTML 5 Web application screen displays an host connection dialog where you can enter the appropriate connection information, complete the host connection, and interact with your green screen host application. This feature gives you access to your host applications without having to open the Design Tool or Web Builder and without having to create and deploy a model.

Using Web Builder project properties for HTML 5 Web applications you can preconfigure a terminal session with a host name and port number. This means you will not be prompted for this information every time you launch the application. This configuration applies to the specific project you are creating and not to the Start menu terminal session links.

1.2 Referencing Web Service Files

Web services are available after deployment in both SOAP and REST formats. Developers can use these documents to identify inputs, outputs, and methods needed to consume the Web service.

You can access your Web service documents here:

List of WSDL documents for deployed models: `http://<session server>:9680/vhi-ws`

List of REST documents for deployed models: `http://<session server>:9680/vhi-rs`

Connect to model (non-pooled): Depending on whether you are accessing a SOAP or REST service - `http://<session server>:9680/vhi-ws/model/<model name>?wsdl` or `http://<session server>:9680/vhi-rs/model/<model name>?json`

Connect to session pool: Depending on whether you are accessing a SOAP or REST service - `http://<session server>:9680/vhi-ws/session/<pool name>?wsdl` or `http://<session server>:9680/vhi-rs/session/<pool name>?json`.

1.3 Referencing API Documentation

API documentation is provided for the Java, JDBC, C, and .NET connectors as well as two additional Host Integrator APIs. See Connectors and APIs for information.

2. Using the Design Tool

The Design Tool is a component of the Developer Kit for modeling and encapsulating an existing host application for integration into client/server and Web applications.

Host Integrator Web Builder automatically generates projects, including Web applications, .NET class libraries, and Java Beans.

2.1 Implementation Options

- What is the host data you want to integrate into another environment or application?

  If at all possible, plan on building a table that represents the host data in a way that can be queried through an SQL statement. Does the data presentation and logic of the host data as presented on the Web differ from the host application itself? If not, a simple rejuvenation may be sufficient.

- Will the host data be presented on a Web page for customers or users?

  What is the Web presentation technology? Know the requirements and the environment where the host data will be used. If you're not integrating the data with other applications, a simple rejuvenation of the host application may be sufficient. Otherwise, you should plan on setting up tables and procedures so that an external application, such as a Web service, has access to the host data. This decision has a direct impact on how your build your model.

- Know the development requirements for the external application.

  Host Integrator provides APIs for integrating host data through SQL, ASP, ASP.NET, JSP, Web Services, COM, and .NET. For a quick implementation, many of these options can be generated using Host Integrator's Web Builder.

You can use Host Integrator with Microsoft’s Active Server Pages (ASP),ASP.NET, or a Java Server Pages (JSP) environment. In this environment, you can use Host Integrator to give users access to host data from a Web browser.

You can use Host Integrator to make host data available to another client/server application, whether or not it is Web-based.

- Host Integrator support for object frameworks on application servers include Web Services, the Component Object Model (COM), and .NET. Application server deployments are especially well-suited for situations in which high volume host systems are extended to the Web. Additional business logic can be developed to add more functionality and capabilities to the system without any work or development required in the host system environment.

2.2 Design Tool Features

- Connecting to a Host
- Using Models to Encapsulate a Host Application
- Abstracting a Host Application
- Providing Core Runtime Services
- Recording Command Lists
- Implementing Preferences
- Providing Online and Offline Design Modes
- Working Collaboratively
- Adding Event Handlers

You can use the following features to capture the functionality of your host application:

2.3 Connecting to a host

The Design Tool can connect to the following hosts:

Host	Models
IBM 3270	Models 2 (24x80) Normal and Extended, 3 (32x80) Normal and Extended, 4 (43x80) Normal and Extended, and 5 (27x132) Normal and Extended
IBM 5250	Models 3179-2, 3180-2 (132 Column Capable), 3196-A1, 3477-FC (132 Column Capable), 3477-FG (132 Column Capable), 3486-BA, 3487-HA (132 Column Capable), 3487-HC (132 Column Capable), 5251-11, and 5291-1
VT	VT102, VT400-7, VT400-8, and VT52 terminals

Host	Models
HP	HP2392A and HP 70092 terminals

You select the host connection type on the New Model or Session Setup dialog box, accessible from the File or Connection menu respectively.

2.4 Using models to encapsulate a host application

The main feature of the Design Tool is the modeling feature, which enables a host expert to create a model of a host application. First, you connect to a host via the Design Tool and then define entities for terminal screens, which may include patterns for identification, attributes to specify the location to input data, and one or more operations to allow programmed traversal of the host application, and variables which can be mapped to various attributes or various entities.

In most cases, you will use tables and procedures to create an abstraction of the host data so that it can be queried through a subset of the industry-standard Structure Query Language (SQL). See "Abstracting a Host Application" below.

The model file `(.modelx)` is saved in the Design Tool and then copied to a Host Integrator Server. For more information, see The Modeling Process.

2.5 Abstracting a host application

You can create procedures (and underlying tables) to add a database abstraction layer on top of your host application model. Client application programmers can then access this database abstraction layer, either by a direct call to a procedure or through a subset of the industry-standard Structured Query Language (SQL). For an SQL query, the client application specifies a table, a set of input parameters, and a set of desired output parameters. Host Integrator then returns the desired data to the client application.

2.6 Providing core runtime services

In addition to the definition process, the Design Tool provides server-like services for the modeling and procedure definitions. This permits the user to test and debug models and database procedures prior to deployment. The model layer requires entity recognition, operation execution, and variable reads and writes. The debug layer takes arbitrary input and resolves a query (or returns an error) or executes a specified query.

2.7 Recording command lists

The command list recorder records host commands for operations required by the debug layer. On the Model menu, point to Record and then click Start Recording to begin. For information about creating login, logout, and move cursor command lists, see Command List Edit.

2.8 Implementing preferences

There are several functional user preferences that can be implemented, including creating default names for attributes, automatic pattern generation, and proposing new operations when appropriate. On the Settings menu, click Preferences for more information.

2.9 Providing online and offline design modes

The Design Tool has the ability to display in online and offline design modes. As each entity is defined, two files are created to enable the design mode process.

- A "screen snapshot," or a snapshot file contains the contents of the display memory and which can then be available to the Design Tool for editing the corresponding entity in offline mode.
- A Host Emulator trace file is a Telnet trace for the terminal screen that the Host Emulator can send over a Telnet connection to simulate the screen being sent from the host.

Offline mode is available for all emulation options, while Host Emulator is available for IBM 3270 and 5250 emulations only.

2.10 Working collaboratively

Host Integrator provides the ability for multiple developers to work similtaneously on a project. Models are saved using the .modelx extension, which gives multiple developers the ability to work collaboratively and merge their changes using a standard version control package, such as Git. When you save a model as a modelx file, all entities, tables, and supporting files are converted into XML, validated by an .xsd file, and saved in the models directory, under the modelx folder. See Working Collaboratively.

You can also import portions of models for use in other models. This makes model creation more efficient. Multiple developers can work on large models simultaneously and pull the various pieces together at a later time. See Importing Model Elements for more information.

2.11 Adding event handlers

An event handler gives you the ability to customize the behavior of a model.

The Design Tool offers a variety of features that assist you in creating event handlers. The result is a Java class that conforms to rules for event handling. This class is then mapped (attached) to specific objects of a model to customize its behavior.

You can attach event handlers to events associated with the entire model, a life cycle event, or to entities, attributes, operations, recordsets and recordset fields, and procedures. You can reuse a handler in multiple models or with multiple objects of the same type within the same model.

If you're ready to begin modeling, see The Modeling Process.

More information

- The Modeling Process
- Working Collaboratively
- Importing Model Elements

3. Planning the Host Integrator Project

3.1 Planning the Host Integrator Project

A model is the representation of a host application's connections, screens, navigation, and data flow that you build with the Verastream Host Integrator Design Tool. Once you have modeled your host application, you deploy the resulting model to a Host Integrator Server, where it can provide real-time access to host data through web-enabled services.

Building a good model of a host application requires a basic understanding of the business problem that will be solved using the Host Integrator model and associated components. Use the guidelines below to prepare for Verastream host application modeling.

- **Gather the basic information about the host and the host application.**

  Know the host operating system and any relevant details about the host application software. The more you know about your host application, the greater the likelihood that you'll have an effective, robust model.

- **Determine the data you want to access in the host application.**

  If you're not familiar with the host application yourself, a host application programmer or an experienced business user needs to identify logon procedures and the sequence of screens typically used to reach the data you want to access. As you build the host application model, each host screen used in the model is represented as an entity. An efficient host application model contains only as many entities as are required to perform the necessary transactions between the external application and the host application.

  Take the time to create an easy-to-follow data map to simplify the process of building the model. This exercise may reveal multiple screens and operations that display the same data. Whenever possible, model the screen that shows the data in its updated form for the transaction in question.

- **Identify the environment where the host data should be available.**

  Will the host data be presented on a web page for customers or users? Is the host data to be used by another application? Know the requirements and the environment where the host data will be used. If you are only planning to update the host application with a more modern web page look and feel, you can build a simple model. However, if you plan to alter the work flow or interoperate directly with other applications, your model will need to be more complex.

- What is the Web presentation technology? Know the requirements and the environment where the host data will be used. If you're not integrating the data with other applications, a simple rejuvenation of the host application may be sufficient. Otherwise, you should plan on setting up tables and procedures so that an external application, such as a Web service, has access to the host data. This decision has a direct impact on how your build your model.

- **Know the development requirements for the external application.**

  Host Integrator provides APIs for integrating host data through SQL, ASP, ASP.NET, JSP, Web Services, COM, and .NET. For a quick implementation, many of these options can be generated using Host Integrator's Web Builder.

- You can use Host Integrator with Microsoft's Active Server Pages (ASP), ASP.NET, or a Java Server Pages (JSP) environment. In this environment, you can use Host Integrator to give users access to host data from a Web browser.
- You can use Host Integrator to make host data available to another client/server application, whether or not it is Web-based.
- Host Integrator support for object frameworks on application servers include Web Services, the Component Object Model (COM), and .NET. Application server deployments are especially well-suited for situations in which high volume host systems are extended to the Web. Additional business logic can be developed to add more functionality and capabilities to the system without any work or development required in the host system environment.
- **Plan for error handling.**

  Error handling is a very important part of the model-planning process. Learn where and how the model can fail or put the host application in an error condition. Then, build exception handling into your model for these cases to reduce the chance that your model will fail after it is deployed.

- **Review the modeling tips in the online help.**

  The modeling tips cover basic model style guide recommendations, a comparison of table-based vs. non-table-based models, basics on screen recognition and pattern usage, and host synchronization.

More information

Using the Design Tool

Learning to use Host Integrator

The Modeling Process

3.2 Host Integrator Components

Host Integrator includes a variety of development and runtime components to integrate host data into Web applications or other client/server applications.

The Development Kit provides a full range of utilities with a limited license server, while the Server Kit includes a full license server without the development tools.

Component	Description	Development Kit	Server Kit
Design Tool	Create a model of the host application by navigating through host screens	Yes	No
Web Builder	Generate web-based applications from the model.	Yes	No
Connectors	Connect to the server to manage host connections and sessions.	Yes	Yes
Session Server	Access data on a variety of host systems.	Yes (limited)	Yes
Web Server	Run Java or HTML5 web application projects.	Yes	Yes (install option)
Host Emulator	Use to test an application without a host connection.	Yes	No
Administrative Console	Use to view and configure server and session information.	Yes	Yes

3.2.1 Development Kit

The Host Integrator Development Kit provides developers with the tools to model and build applications that integrate legacy application information. It includes the following components:

- Design Tool
- Web Builder
- Connectors
- Server (limited-license version)

- Administrative Console
- Log Utilities
- Host Emulator
- Verastream Help

3.2.2 Server Kit

The Host Integrator Server Kit is a suite of applications that allow you to deploy applications created with the Host Integrator Development Kit. It includes the following components:

- Server (full-license version)
- Administrative Console
- Connectors
- Log Utilities
- Verastream Help

3.2.3 Adding, Removing, and Repairing Components

You can add components to or remove components from your Host Integrator installation whenever it becomes necessary. You can also repair your existing installation using the same install program.

1. Navigate to `<install_directory>/setup` and run `setup.exe`. The install program will detect the existing installation and display the *Maintenance and Component Selection* panel. By default the Maintenance tab displays.
2. Repair - Reinstalls missing or corrupt files, registry keys, and shortcuts. Settings may be reset to their default values.

Reinstall - Completely removes from the system and begins the installation process anew. The Install Guide, in PDF format, is available from your installation directory.

4. Remove - Removes all Host Integrator files and directories as well as uninstalls all product components.

   You cannot continue with your installation maintenance selection (repair, reinstall, remove) if you click Continue and navigate away from the Maintenance tab. Select the maintenance option and click Continue while on the Maintenance tab to perform the desired action.

5. To view a list of currently installed components, open the *Component Selection* tab. This tab displays the components that are currently installed. Select or clear components to create the desired installation.
6. Click *Continue* and setup will perform the necessary actions to install or remove your selections.

3.2.4 Starting and Stopping Services

There are several Host Integrator services. Different services are installed depending on your platform, type of product (Server or Developer Kit), and other options that you select during installation.

Host Integrator services include:

- Session Servers

  Runtime engine that loads deployed models; connects to host systems, and services request from clients.

- Log Manager

  Captures runtime messages in log files; and handles SNMP and email notification.

- Management Server

  Supports the Administative Console, session server load distribution domains, authentication and authorization security, and session pool scheduling.

- Web Server

  Runs Java and HTML5 Web projects generated by Web Builder.

- Host Emulator

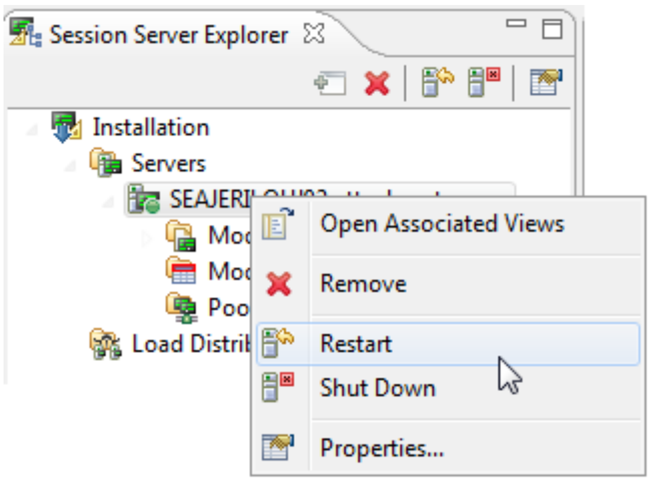
Simulates a host (3270 or 5250 only) for demonstration, training, or testing purposes.

On Windows, with the Developer Kit installed, the services start automatically on system startup. When you install the Server Kit, both the Host Emulator and Web Server are disabled at startup. For Linux or zLinux see the install guide for instructions on how to configure the system to start services automatically.

To start or stop services from the Administrative Console

The Session Server, Management Server, and Host Emulator services running on Windows or Linux servers can be restarted or stopped using the Administrative Console.

Open the Administrative Console,from each of the server Explorer panes, select the server you want to interact with, right-click and choose Restart or Shut Down.



To start or stop services from the command line

Commands or shell scripts can be run at a shell command prompt, called from other scripts, or run from a shortcut that you create.

- Windows

  Navigate to `<install_directory>/HostIntegrator or/bin/services` directory. You can start, restart, or stop all services by running the appropriate batch file or by interacting with a specific service. To interact with a specific service, open the component directory, for example HostEmulator, and run the batch file for the action you want to perform; `Restart Host Emulator.bat, Start Host Emulator.bat, or Stop Host Emulator.bat`.

- zLinux

  Navigate to `<install_directory>/opt/rocketsoftware/verastream/HostIntegrator/bin` directory. You can start, restart, or stop all services by running the appropriate script file or by interacitng with a specific service. For example, to restart all services, run `atstart -restart all` or to restart just the management server, run `atstart -restart mgmtserver`.

More information

The Development Process

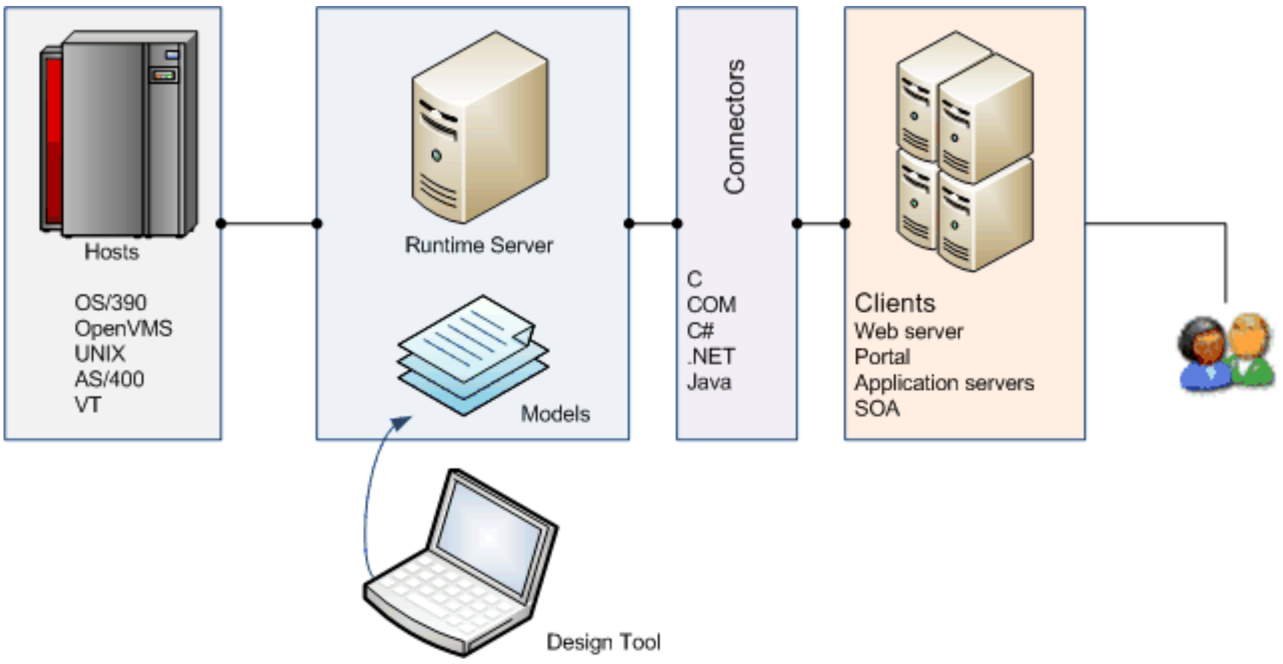
Planning the Project

3.3 Development Process

To build a Host Integrator application, follow these steps:

1. Plan your integration project

   To create a good model of a host application you must have a basic understanding of the business problem that you want the model to solve. There are guidelines available to help you plan your project in Planning the Project.



2. Create and deploy the host application model

   There are a few steps you must follow to create and deploy a model.

   Navigate through the host application, identifying the screens and fields that lead to the data that you want to integrate into the new Web or client/server application.

   Expose the resulting components as database tables and procedures. A table is a database abstraction with attributes or recordset fields structured in table columns. You can create a procedure that defines how Host Integrator locates, retrieves, updates, inserts, or deletes the data when it fulfills a request through SQL or through another API. If you want to add functionality beyond what is provided in the Design Tool, you can add event handlers that extend or override default model behavior.

Test your model to confirm that it operates as you expect.

Deploy the model to a folder containing the model file and any supporting files on the Host Integrator server. Web services are automatically provided by the session server as an embedded SOAP stack or as a REST service after a model package is deployed using the Design Tool. The embedded Web service supports all model procedures and features, including executeSQLStatement and ProcessString event handlers. See About Verastream Web Services for detailed information on Host Integrator Web services.

3. Build a new Web or client/server application

You can build a new application using these options:

Use Web Builder to quickly build and deploy a Web application. You can build a Web application for a Java, .NET, or ASP interface, and then launch a standard editor for any of these interfaces to extend or modify the Web application.

Use Web Builder to generate a component interface that provides access to the procedures and screens in a host model. Component interfaces are available using Java Beans or .NET 2.0 class libraries.

Use the IDE of your choice to consume Verastream-generated components. Mix and match components or services to create composite applications. You can use the Host Integrator connectors to develop a new client/server or Web application. Using these APIs, a developer can write a client/server or Web application without extensive knowledge of how the host application works.

4. Deploy the application to the enterprise

In a production environment, the deployed Host Integrator servers provide dynamic load balancing and failover in transaction-intensive environments. Host Integrator supports all mainframe-based security packages and also has its own multi-level security, which can be tied to the security schema of the selected deployment platform (Windows or Linux).

5. Using the Administrative Console

The rich Administrative Console user interface keeps mass management in mind and provides central administration of the management server and Host Integrator servers. It is here you will be able to:

Maintain control over sessions and pools

Monitor logging of session information, activity, use and status

Configure external reporting

Configure SNMP and JMX support for third party consoles

Manage models and edit model properties

3.4 Learn to Use Host Integrator

To get the most benefit from this tutorial, review the following topics before you begin:

- Using the Design Tool
- The Development Process

You will need to install the Developer Kit to run this tutorial.

3.4.1 Goal

The goal of this tutorial is to produce a Host Integrator application.

You will work with a simulation of a 3270 (CICSAccts) host application and use Verastream tools and utilities to build a simple Host Integrator application. This is a demonstration application, consisting of limited navigation and a database of generic customer data for eight fictitious customers. It is a useful tool for learning to create a model.

This tutorial walks you through these steps:

1. Define the business need for the project
2. Develop the model requirements and map the data
3. Build, deploy, and test the model and web service
4. Creating a Web Builder project and generating a Web application
5. Deploy the project to the enterprise

Let's get started.

3.4.2 Defining the Business Need

Often the business needs of a project have already been defined before the development process ever begins. Although you may not have been a part of the decision, understanding the business problem is important to building a successful application.

In this CICSAccts project, the business goal is to make specific customer information available to users over the Internet and to other business processes using Web services and other programmatic component interfaces. You will access the data through the CICSAccts application on a 3270 host.

To accomplish the goal you will: automate the login to the host, retrieve the host data, and display the data in a simple Web application.

3.4.3 Developing Model Requirements and Mapping the Data

Now that you understand the goal and the deliverables of the project, you are ready to use the Host Integrator Design Tool to build a model of the application. The model contains all the information required for host application navigation, screen recognition, data storing, and data retrieval.

During this phase, either working on your own or with someone familiar with the host application, you will navigate through the application and document each host screen's inputs, outputs, navigation path, and other quirks. This will result in a model requirements document that identifies the intricacies of each of the host screens that will be required to access the data.

The model requirements document contains two important features: the list of entities (unique host application screens) that need to be modeled, and a chart mapping the project data to the screen where the data will be retrieved.

In our application, in order to retrieve the necessary data, the main screen in the CICSAccts application needs to be treated as two distinct entities: One for input of information to do the search (Main) and one for output of the search (NameSearchResults). The output is a listing of information about each customer; this is the data that you need from the application.

This is a map of the data our model needs for inputs and outputs:

Attribute Name	Used for	Screen	Screen Field Name
LastName	input	Main	SURNAME
FirstName	input	Main	FIRST NAME
AcctNum	input	Main	ACCOUNT
RequestType	input	Main	REQUEST TYPE
LastName	output	NameSearchResults	SURNAME
FirstName	output	NameSearchResults	FIRST
MiddleInitial	output	NameSearchResults	MI
Title	output	NameSearchResults	TTL
Address1	output	NameSearchResults	ADDRESS
AcctNum	output	NameSearchResults	ACCT
Reason	output	NameSearchResults	ST

Attribute Name	Used for	Screen	Screen Field Name
ChargeLimit	output	NameSearchResults	LIMIT

3.4.4 Building and Deploying a Simple Model

The completed model of the CICSAccts application lets you query the CICS database by a customer's last name. To build this model, you will:

Connect to host using a login command list

Create entities from host screens

Create operations for host navigation

Create recordsets for scrolling data

Create tables and procedures for abstraction level queries

Create procedures to retrieve data

Test and deploy the model

Connect to Host Using a Login Command List

Open the Host Integrator Design Tool.

Click **File** > **New** to open the New Model dialog box.

Type `MyModel` in the **Model Name** field. This creates a connection profile to the CICS host. Accept the default model location.

4. In the **Settings box**, click **Browse** and select **IBM3270Model**. This provides the appropriate default settings for the session type (IBM 3270 Terminal), terminal ID (Model 2 24X80 Extended), and transport type (Telnet). The CICS demonstration host uses port 1097.

   Enter the following host connection information:

Field Name	Entry
Host name or IP address	localhost
Port	1097
Device	(Leave blank)

5. From the **Settings** menu, select **Preferences**. In the Preferences Setup dialog box, select the option **When model file opened connect to host**. This configures the Host Integrator Design Tool to connect to the host automatically when the model is opened.

Click **File** > **Save MyModel.modelx**.

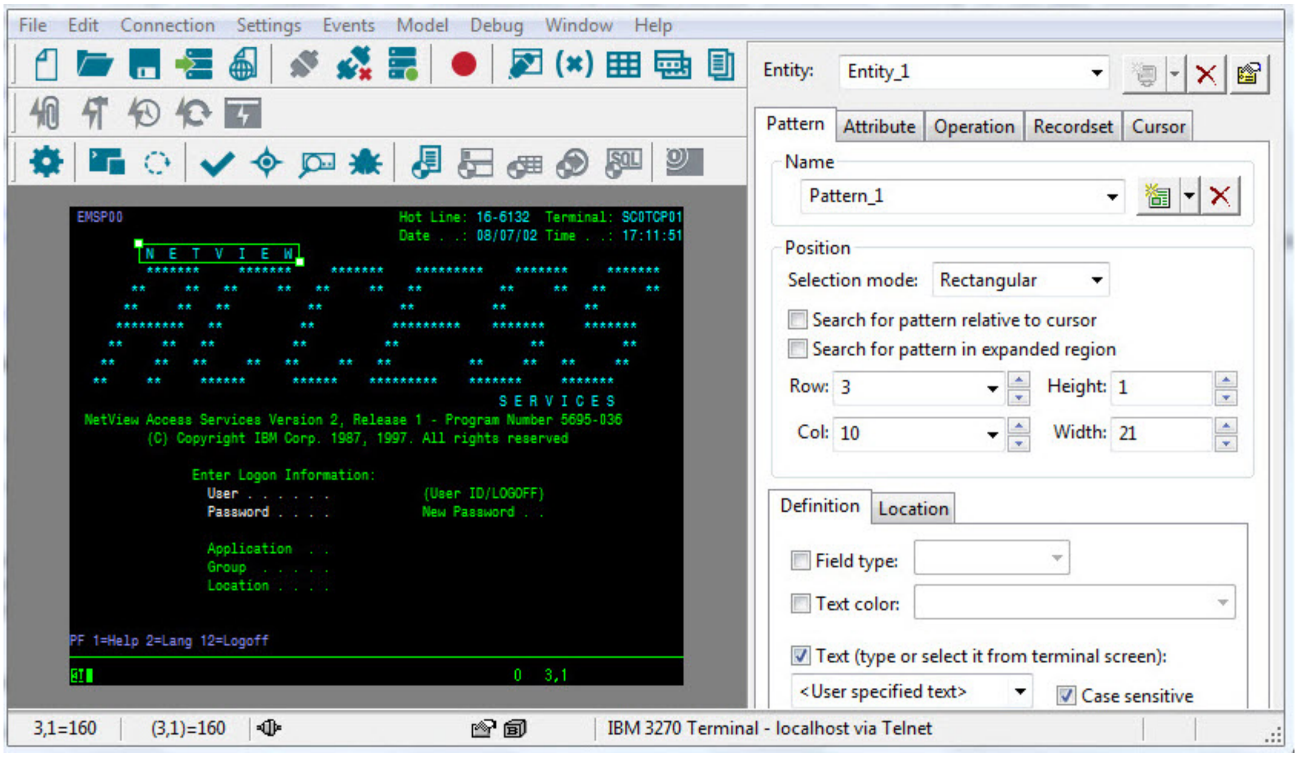
You have now finished configuring the Design Tool to connect to the demonstration 3270 application (CICSAccts) running in the Host Emulator on the local machine over port 1097. Every time a new entity is created (host screen is accessed), the Design Tool automatically notes all operations (navigation through the host application). You will learn more about these terms and processes as you proceed through this demonstration.

About the Design Tool

The Design Tool is divided vertically into two sections or panes:

Terminal Window -- The left pane displays the actual host application screen.

- Entity Window -- The right pane contains the settings used to define the host screens that make up the model.



```
EMSP00                                  Hot Line: 16-6132  Terminal: SCOTCP01
                                        Date . .: 08/07/02 Time . .: 17:11:51
          N E T V I E W
           *******   *******    *******   *********   *******   *******
         **    **  **    **  **    **  **          **    **  **    **
        **    **  **          **          **          **          **
       *********  **         **        *********   *******   *******
      **    **  **         **         **                **         **
     **    **  **    **  **    **  **          **    **  **    **
    **    **    ******     ******    *********   *******    *******
                                                         S E R V I C E S
 NetView Access Services Version 2, Release 1 - Program Number 5695-036
         (C) Copyright IBM Corp. 1987, 1997. All rights reserved

               Enter Logon Information:
                 User . . . . . .              (User ID/LOGOFF)
                 Password . . . .              New Password . .

                 Application  . .
                 Group  . . . . .
                 Location . . . .


PF 1=Help 2=Lang 12=Logoff
```

```
4I                                                     0   3,1
```

Entity: Entity_1

Name: Pattern_1

Position

Selection mode: Rectangular

☐ Search for pattern relative to cursor

☐ Search for pattern in expanded region

Row: 3 Height: 1

Col: 10 Width: 21

☐ Field type:

☐ Text color:

☑ Text (type or select it from terminal screen):

<User specified text> ☑ Case sensitive

3,1=160 (3,1)=160 IBM 3270 Terminal - localhost via Telnet

CICS application navigation tips

Use these navigation tips to negotiate the CICSAccts application example:

The Page Down key on your keyboard is mapped to the host PA2 (page down) key.

The Scroll Lock key on your keyboard is mapped to the host CLR (clear) key.

Pressing **Enter** returns you to the main screen.

Creating a Login Command List

A login command list provides a quick and reliable login to a host. You can also use login commands to bypass a host login screen.

To create a command list to log on to CICS:

If you're not already connected, click to connect to the model.

Click **Model** > **Record** > **Start Recording**.

Click in the Terminal Window and then press the Scroll Lock key (this is mapped to the host CLR by default).

In the terminal window, type `accts`, and press **Enter**.

Click **Model** > **Record** > **Stop Recording**.

Select the **Save as login command list** option and click **Save**. The Command List Edit dialog box opens. You will use this dialog box to edit the login command list.

Tip

To open the Command List Edit dialog box, click **Model** > **Properties**, and then .

Editing the Login Command List

The CICSAccts demonstration application is run from the Host Emulator; to synchronize the Host Emulator login to the application, you need to add an additional wait command to the top of this recorded command list.

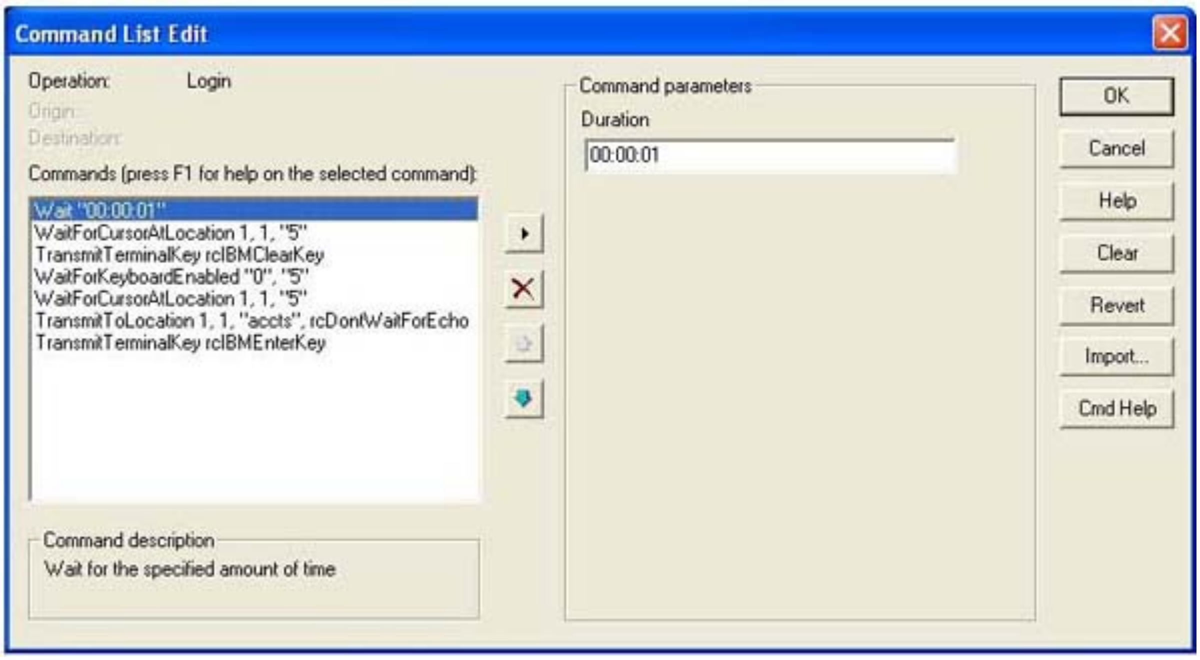
To edit the login command list:

Select the first command in the **Commands** field, `WaitForCursorAtLocation, 1, 1, "5"`. If the Row and Column values are not both 1, change the values to *1*.

Click **Insert** , then point to **Host Events**, and click **Wait**.

With **Wait "00:00:01"** selected, click **Move Up** to move this command to the top of the command list. Your commands should look like the command list in Figure 2.

4. Click **OK** to close both the Command List Edit and Model Properties dialog boxes.



Operation: Login
Origin:
Destination:

Commands (press F1 for help on the selected command):

```
Wait "00:00:01"
WaitForCursorAtLocation 1, 1, "5"
TransmitTerminalKey rcIBMClearKey
WaitForKeyboardEnabled "0", "5"
WaitForCursorAtLocation 1, 1, "5"
TransmitToLocation 1, 1, "accts", rcDontWaitForEcho
TransmitTerminalKey rcIBMEnterKey
```

Command description
Wait for the specified amount of time

Command parameters
Duration: 00:00:01

Configuring the Login Command List to Load Automatically

The Design Tool automatically executes this login command list whenever you connect to the model.

To configure the login command list to load on connection:

Click **Settings** > **Preferences**.

In the Preferences Setup dialog box, on the **General** tab, select **Execute login commands**, and click **OK**.

Save your model.

To test the login command list:

On the tool bar, click to disconnect your current session.

Click to reconnect and run the login command list. Each time you open the model, the Design Tool automatically connects and logs you onto the host and displays the Main screen of the host application.

Create Entities From Host Screens

An entity is usually a single host application screen, such as the login screen or the main menu screen of an application. A model is made up of entities (host screens) and the attributes, recordsets, and operations associated with those screens.

Up to this point, you have recorded all navigation through the host application in the login command list. Since the end application does not access data from these initial screens, you do not have to configure each screen in the login sequence as an entity. The first screen you need to define is the Account File Menu screen.

To add a screen as an entity:

When the first screen, Account File: Menu, appears after connecting to the host, click **New Entity** in the Entity Window. By default, the entity is named `Entity_1`.

Change the entity name to `Main`.

Creating a Pattern for Entity Recognition

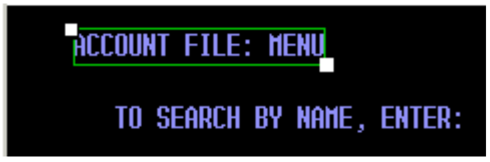
A pattern is a model element that is used in the screen signature to identify a particular host application screen. Host Integrator uses patterns to distinguish individual entities. Select any static text on the screen, such as field labels, or screen headers and footers, to be designated as a pattern. Each pattern can include up to 259 characters.

Tip

To help ensure that each entity has a unique identifier, it is a good idea to configure at least two patterns to define each entity, one at the top of the screen and one at the bottom.

To create patterns for the Main entity:

1. In the Terminal Window, use the mouse to select the text: `ACCOUNT FILE: MENU`



```
ACCOUNT FILE: MENU

    TO SEARCH BY NAME, ENTER:
```

2. On the Pattern tab, click **New Pattern**.

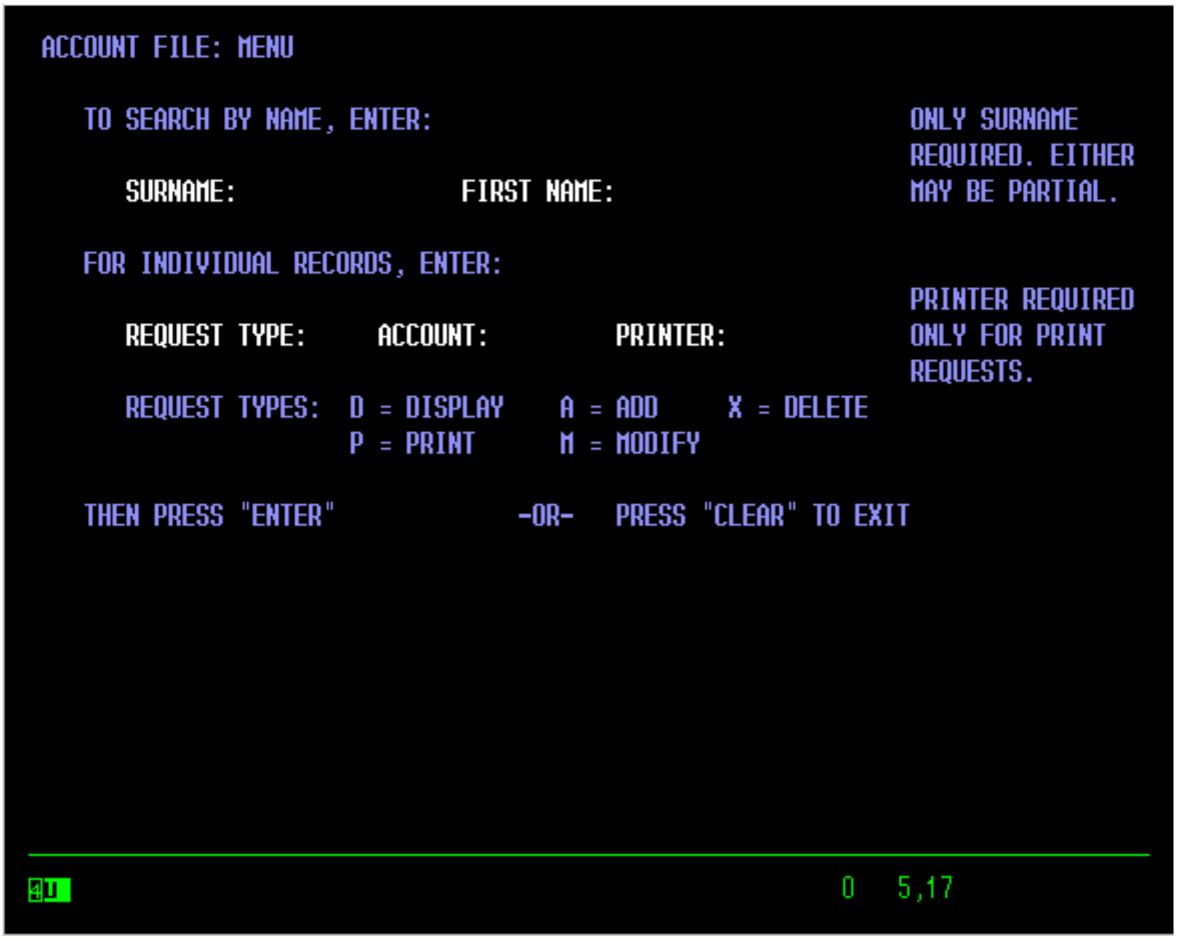
By default, the pattern is named `Pattern\_1`. Accept this default name.

Click **Apply**.

The Design Tool has now recorded the position, properties, and signature parameters of the pattern. You can access and edit these settings on the Pattern tab.

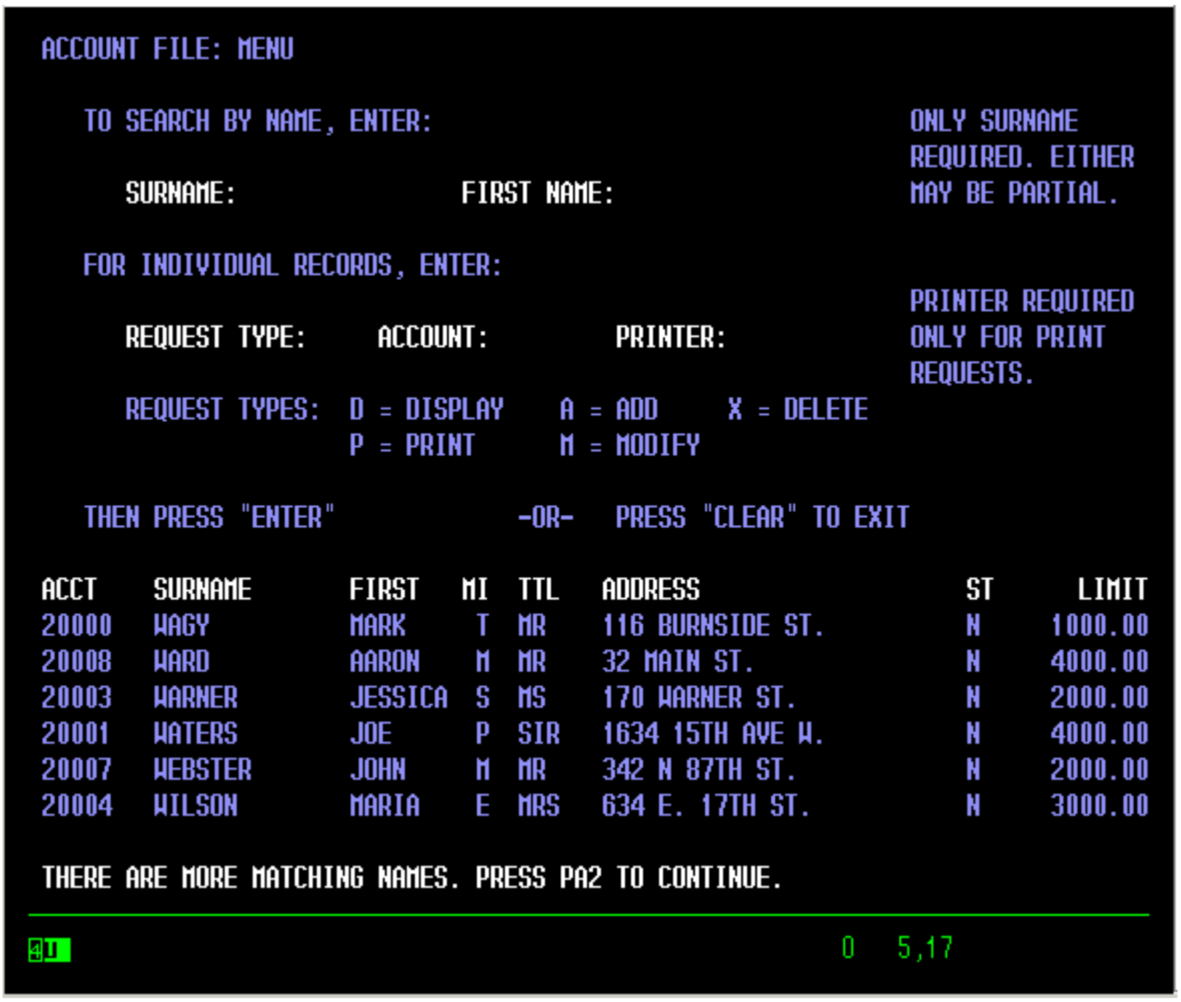
Typically, it is a good practice to define at least two patterns per entity, and on most host screens you would select a second string of unique text to use as your second pattern. However, when you defined the requirements for the model, you realized that for this application you must take a different approach because the same host screen is used to both request and display customer information.

When you enter customer information on the screen (Figure 4), the data is returned in the blank area at the bottom of the screen (Figure 5). Compare Figure 4 to Figure 5; all of the text on the top half of the screen remains the same. It is difficult to assign a unique pattern on the first screen because all the static text on the first screen also appears on the second screen.



```
ACCOUNT FILE: MENU

   TO SEARCH BY NAME, ENTER:                           ONLY SURNAME
                                                       REQUIRED. EITHER
      SURNAME:              FIRST NAME:                MAY BE PARTIAL.

   FOR INDIVIDUAL RECORDS, ENTER:
                                                       PRINTER REQUIRED
      REQUEST TYPE:    ACCOUNT:        PRINTER:        ONLY FOR PRINT
                                                       REQUESTS.
      REQUEST TYPES:  D = DISPLAY    A = ADD     X = DELETE
                      P = PRINT      M = MODIFY

   THEN PRESS "ENTER"            -OR-   PRESS "CLEAR" TO EXIT


4U                                                   0   5,17
```

Figure 4 -- Entity 2 \"main\"



```
ACCOUNT FILE: MENU

   TO SEARCH BY NAME, ENTER:                                   ONLY SURNAME
                                                               REQUIRED. EITHER
      SURNAME:                FIRST NAME:                      MAY BE PARTIAL.

   FOR INDIVIDUAL RECORDS, ENTER:
                                                               PRINTER REQUIRED
      REQUEST TYPE:    ACCOUNT:          PRINTER:              ONLY FOR PRINT
                                                               REQUESTS.
      REQUEST TYPES:  D = DISPLAY    A = ADD     X = DELETE
                      P = PRINT      M = MODIFY

   THEN PRESS "ENTER"             -OR-   PRESS "CLEAR" TO EXIT

ACCT    SURNAME       FIRST   MI  TTL   ADDRESS                  ST      LIMIT
20000   WAGY          MARK     T  MR    116 BURNSIDE ST.         N     1000.00
20008   WARD          AARON    M  MR    32 MAIN ST.              N     4000.00
20003   WARNER        JESSICA  S  MS    170 WARNER ST.           N     2000.00
20001   WATERS        JOE      P  SIR   1634 15TH AVE W.         N     4000.00
20007   WEBSTER       JOHN     M  MR    342 N 87TH ST.           N     2000.00
20004   WILSON        MARIA    E  MRS   634 E. 17TH ST.          N     3000.00

THERE ARE MORE MATCHING NAMES. PRESS PA2 TO CONTINUE.

4B                                                   0   5,17
```

Figure 5 -- Entity 2 \"NameSearchResults\"

Can you treat this as one entity? You could, but that would add complications when creating operations and recordsets.

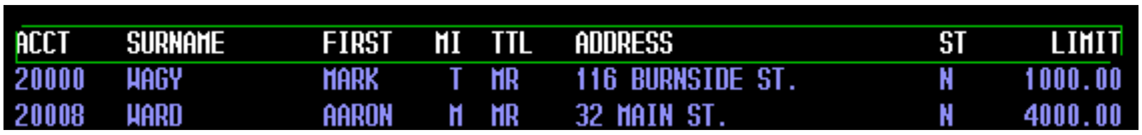
The simplest solution is to create two entities and use an exclude; Host Integrator "recognizes" the first entity by the presence of specific text, and the second by the absence of that same text.

To use an exclude, select a pattern that exists only on the second screen, and then configure Design Tool to recognize the first screen by the absence of that pattern. In other words, when the pattern is not there, then by default, the first screen is being accessed.

To create a pattern using an exclude:

In the Terminal Window, click in the **SURNAME** input area, type `W` (uppercase W), and then Press **Enter**. You are now on the second entity (the same screen as the first entity, but with data).

2. Use the mouse to select the entire account information heading:



```
ACCT     SURNAME        FIRST    MI  TTL   ADDRESS                    ST       LIMIT
20000    WAGY           MARK      T  MR    116 BURNSIDE ST.           N      1000.00
20008    WARD           AARON     M  MR    32 MAIN ST.                N      4000.00
```

Figure 6 -- Select this text to create a second pattern

3. On the **Pattern** tab, click the **New Pattern** button. Accept the default pattern name, `Pattern\_2`.

At the bottom of the **Pattern** tab, select **Screen properties not present** as part of the Definition and click **Apply**.

Click **Yes** to dismiss the warning message box.

When you use the *Screen properties not present* setting, Design Tool automatically initiates the creation of a new entity. Before continuing, you must complete the steps to add this second entity and identify it with patterns.

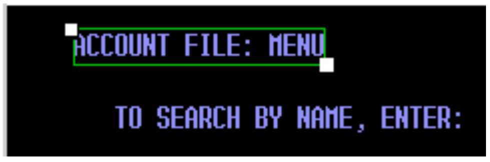
To add this screen (with data) as a new entity:

Click the **New Entity** button in the Entity Window.

Select the default entity name (`Entity\_2`) and replace it with `NameSearchResults`.

To add patterns for the second entity, NameSearchResults:

1. In the Terminal Window, use the mouse to select the text: `ACCOUNT FILE: MENU`

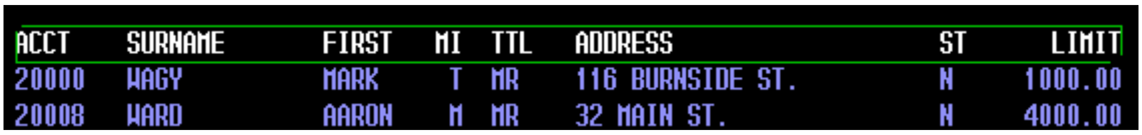


```
ACCOUNT FILE: MENU

     TO SEARCH BY NAME, ENTER:
```

Figure 7 - Select this text in the Terminal Window to create a pattern in the second entity

2. On the **Pattern** tab, click the **New Pattern** button. Accept the default name.

3. Use the mouse to select the entire account information heading:



```
ACCT     SURNAME        FIRST    MI  TTL   ADDRESS                    ST       LIMIT
20000    WAGY           MARK      T  MR    116 BURNSIDE ST.           N      1000.00
20008    WARD           AARON     M  MR    32 MAIN ST.                N      4000.00
```

Figure 8 - Select this text to create a second pattern in the second entity

This is the same text you used when you identified the Main entity; however, in that instance you configured the Design Tool to make sure that the pattern was not present when identifying the Main entity. In identifying the NameSearchResults entity, you are configuring the Design Tool to verify the pattern is present.

4. On the Pattern tab, click the **New Pattern** button. Accept the default name and click **Apply**.

On the Confirm Operation dialog box, click **Discard**. Then click **Yes** to confirm the discard and close the dialog box. (Design Tool automatically creates navigational operations; however, you need to first identify attributes and fields.)

You have now created two entities, **Main** and **NameSearchResults**.

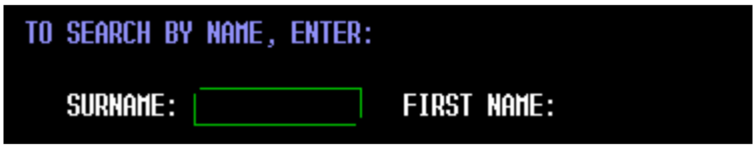
The next step is to identify the fields or attributes that you need on each entity. To return to the Main entity, click **Disconnect** on the toolbar, and then **Connect** to reconnect to the host, run the login script, and open the Main entity.

Configure Attributes for Data Entry and Retrieval

An attribute is a selected area on an entity that contains data that is accessible through the model. This area might be a data entry field or a text field that changes depending on the choices you make on prior screens. You use attributes to get and set data on entities. Create attributes only for those host fields that need to be accessed; give attributes meaningful names that identify their purpose.

To create attributes:

1. On the **Main** entity, select the **SURNAME** field input area.(On block mode applications such as this 3270 application, you can double-click the field to select it.)



TO SEARCH BY NAME, ENTER:

SURNAME: FIRST NAME:

Figure 9 -- The defined area of Attribute_1

2. On the **Attribute** tab, click **New Attribute** to add this as `Attribute_1`.
3. Rename `Attribute_1` to `LastName` and if necessary:

   Adjust the start position to row 5, column 17.

   Adjust the end position to height 1, width 12.
4. Repeat this process to include these attributes:

Attribute	Attribute Name	Start (row,column)	End (height, width)
SURNAME	LastName	5,17	1,12
FIRST NAME	FirstName	5,44	1,7
REQUEST TYPE	RequestType	9,22	1,1

Attribute	Attribute Name	Start (row,column)	End (height, width)
ACCOUNT	AcctNum	9,35	1,5

5. Click **Apply**.

Next, add these same attributes to the **NameSearchResults** entity:

To add the attributes:

1. To navigate to the **NameSearchResults** entity, on the **Main** entity, click the **SURNAME** input field and type `W` (uppercase). Then press **Enter**.

   A Confirm Operation dialog box displays listing the commands needed to navigate to the NameSearchResults entity.

2. Click **Approve**.

Repeat the steps to add the same four attributes (**LastName, FirstName, RequestType, AcctNum**) on the **NameSearchResults** entity.

Click **Apply** and save your model.

Create Operations for Host Navigation

Operations are typically used to navigate between entities, so they have an origin entity and a destination entity. You've already configured the Design Tool to auto-generate operations as you move through the host application. When you navigated from the Main entity to NameSearchResults, you approved the operation `[ToNameSearchResults]{.uielements}`.

So far, you have only navigated forward through the application. To move backwards by selecting the prior entity you need to create a return operation. This ensures entities can navigate both forwards and backwards in a model.

Open the drop-down list of available entities. The icon preceding the Main entity selection Main indicates the home entity. A red house indicates that you have not defined operations to access this entity from your current location.

To create a return operation:

Click in the Terminal Window and press **Enter**.

In the Confirm Operation dialog box, review the new operation, **ToMain**, and then click **Approve**.

To review the navigation operations, open the drop-down list in the Entity box. Both entity icons are now available. Select each entity and view the entity's Operation tab.

Tip

If you click random keys while navigating, each key press is recorded in the operation. If you see unnecessary operations, use the Delete button to remove them from your operation command list.

Create Recordsets for Scrolling Data

Recordsets handle scrolling or tabular data. You create recordsets to manage dynamically changing data that spans several host screens. In the Design Tool, you create a recordset by defining the:

Position and layout of the recordset on the screen

Methods for scrolling through the recordset data

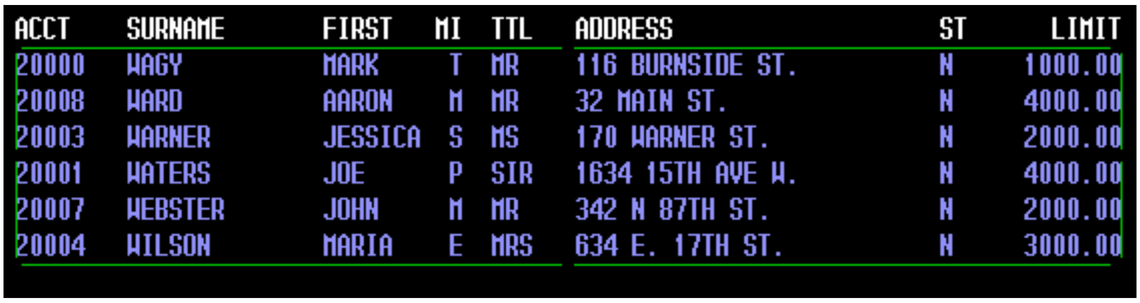
Fields of data in the recordset

Defining the position and layout of the recordset

You are going to create a recordset from the data displayed in the list of records on the NameSearchResults entity.

Select the **NameSearchResults** entity and click the **Recordset** tab.

2. With the mouse, select the full scrollable area on the screen. Do not include the column titles when you select the scrollable region.



```
ACCT   SURNAME     FIRST    MI  TTL  ADDRESS              ST     LIMIT
20000  WAGY        MARK      T  MR   116 BURNSIDE ST.     N    1000.00
20008  WARD        AARON     M  MR   32 MAIN ST.          N    4000.00
20003  WARNER      JESSICA   S  MS   170 WARNER ST.       N    2000.00
20001  WATERS      JOE       P  SIR  1634 15TH AVE W.     N    4000.00
20007  WEBSTER     JOHN      M  MR   342 N 87TH ST.       N    2000.00
20004  WILSON      MARIA     E  MRS  634 E. 17TH ST.      N    3000.00
```

Figure 10 -- The scrolling region of this screen is selected. When selected, the lines around this area of your screen are green

3. Click **New Recordset** . The **Position** tab should show the **Top** of the selection at Row 17, Column 2, and the **Bottom** at Fixed row, Row 22.

By default, the recordset is named `Recordset_1`. Change the name to `AccountList` in the **Name** box and then click **Apply**.

Defining the scrolling operation through the recordset data

You view the recordset data by using terminal keys to page down and scroll through the data. Navigational operations are defined within each recordset.

Adding a page-down operation

You'll need to define a page-down operation for the **AccountList** recordset in the **NameSearchResults** entity.

To define a page-down operation for the AccountList recordset:

Select the **Operation** tab of the **NameSearchResults** entity.

To record your operation, from the **Model** menu, choose **Record > Start Recording**.

Click in the Terminal Window and press the Page Down key, which is mapped to host key PA2.

Click **Model > Record > Stop Recording**.

In the **Stop Recording** dialog box, rename the operation from `ToNameSearchResults` to `PageDown` and click **Save**.

6. In Step 3, several additional commands may have been recorded. This operation only requires these two commands:

```
CheckOperationConditions
TransmitTerminalKey rcIBMPA2Key
```

   If other commands are listed, delete them.

7. Click **OK** to save and close the Operation Edit dialog box.

Using the page-down operation

After you create the recordset operation, you must tell Host Integrator when to use it. The Recordset Operations dialog box contains a predefined set of procedures that the Design Tool uses for specific operations. A page-down procedure displays an entire page of new data. You need to associate a specific operation with this page-down procedure.

To associate a specific operation with a procedure

From the **Recordset** tab, click the **Operations** button at the bottom left of the tab.

On the Recordset Operations dialog box, open the **PageDown** drop-down list.

Select your PageDown operation from the list of options.

Click **Close**, and then click **Apply**.

Terminating the page-down operation

Next, you need to communicate to Host Integrator when to stop scrolling through the NameSearchResults data. To do this, you first create a pattern so Host Integrator recognizes that the end of the data has been reached. You can often use a comment such as *END OF DATA* for this.

Unfortunately, the CICS application does not display an end-of-data comment. Instead, you can configure Host Integrator to recognize the absence of the text, "THERE ARE MORE MATCHING NAMES," as an indication that it has reached the end.

To create the pattern

On the **NameSearchResults** entity, open the **Pattern** tab.

Click in the Terminal Window and press **Enter**.

In the **Surname** field, type `W` (uppercase W), and press **Enter**.

Select the text: `THERE ARE MORE MATCHING NAMES` and then click the **New Pattern** button.

Rename the pattern to `MoreNames`.

Clear the **Use in entity signature** check box.

Select **Screen properties not present**.

Click **Apply**.

Now that you have created the pattern, you must tell Host Integrator to use the MoreNames pattern to signal the end of the PageDown operation.

To signal the end of the PageDown operation

Open the **Recordset** tab and click the **Termination** button at the bottom of the tab.

Open the **Scroll Down** tab, clear the **Scroll operation results in same recordset data** check box in the **Scroll termination criteria** pane.

Select **Screen contains pattern**, and then from the drop-down list, select the **MoreNames** pattern.

To configure the **Exclude** records, on the last screen option select **Read until __ blank records are found**. Then accept the default setting of `1`.

Click **Close**, and then **Apply**.

You can now successfully traverse and terminate the AccountList recordset.

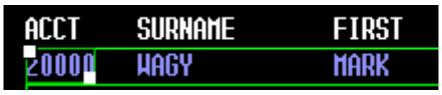
Creating Recordset Fields

A field is a single piece of data in a recordset. Defining a field on a recordset is similar to defining an attribute on an entity. For the CICSAccts application, you need the customer's first and last name, middle initial, title, and address, as well as the reason and charge limit on their account.

To add fields to the AccountList recordset

1. From the **Recordset** tab, open the **Fields** tab.

2. In the Terminal Window, select **account number 20000** on the first row of data (the first row of data is used by default to define what data is contained in this column):



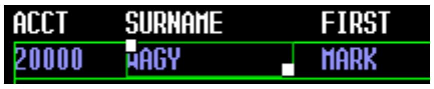
```
ACCT    SURNAME        FIRST
20000   WAGY           MARK
```

Figure 11 -- Defining the AcctNum field

3. On the **Fields** tab, click **New Field** .

   A new field called `Rfield_1` is created.

4. Rename the new field to `AcctNum`.
5. Select the entire surname field on the first row of the recordset. Then click the **New Field** button and rename it to `LastName`.



```
ACCT    SURNAME        FIRST
20000   WAGY           MARK
```

Figure 12 -- Defining the Surname field

6. Use the data below to add and define the remaining fields.

Select	Default Name	Rename to	Start	End
WAGY	RField_2	LastName	9	20
MARK	RField_3	FirstName	23	29
T	RField_4	MiddleInitial	32	32
MR	RField_5	Title	35	38
116 BURNSIDE ST	RField_6	Address1	67	67
N	RField_7	Reason	67	67
1000.00	RField_8	ChargeLimit	72	79

7. Click **Apply** and save your model.

Create Tables and Procedures for Abstraction Level Queries

You have now created a model of your host application that defines host screens as entities, screen data as attributes, and tabular data as recordset fields. You have also created operations that describe how to navigate from one entity to another. At this stage of the process, the model fully describes the host application layout.

Tables and procedures are functionally interlocked; tables are populated with data by a procedure, and you can access the unstructured data in a way that resembles a structured database table.

Because Host Integrator tables do not contain data, but rather "organize" host data into a database-like view, the only way to access abstracted table data is through a procedure. Procedures define how Host Integrator locates, retrieves, updates, inserts, or deletes data when it fulfills a request from the client application.

The Web application developer needs to know only which procedure to run, along with the appropriate inputs and outputs; there's no need to know how the host application works. For more information, see Tables Overview.

To create a table

1. From the **Model** menu, click **Tables**.
2. On the Introduction dialog box, select **Don't show me this dialog again** and click **Finish**. You will not be using the Table Wizard to create this table.
3. On the **Tables** dialog box, click **New**. On the **Create a New Table or Procedure** dialog box, select **Table** and click **OK**.
4. In the **Name** field, change Table_1 to `Accounts`.
5. In the **Description** field, type `Table of account information`.
6. Click **Insert Column** six times, to create Column 1 through Column 6. Columns identify the attributes and recordset fields that make up the table.
7. Using the following chart, rename Column_1 through Column_6. For example, rename Column_1 to `LastName` and so forth. You do not need to enter data in any of the other fields.

Default Name	Rename to
Column_1	LastName
Column_2	FirstName
Column_3	MiddleInitial
Column_4	Title
Column_5	Address1
Column_6	AcctNum

8. Click **Apply** to save the **Accounts** table.

Create Procedures to Retrieve Data

Procedures retrieve data and populate tables.

To create a procedure that retrieves the account information

1. In the Tables dialog box, click **New** and select **Procedure** (using the Procedure Wizard). Then click **OK**.
2. In the **Name** field of the Procedure Wizard, type `SearchByName`.
3. In the **Description** field, type `Search by name`. Then verify that the **Procedure type** option is `Select` and click **Next**.
4. Identify the procedure parameter needed to manipulate and return host data and specify which attribute or field to use to filter the records:

   a. In the Filter Parameters dialog box, go to the `LastName_ row` and click in the empty **Write parameter to** cell.

   b. Open the drop-down list and select **Main.LastName** to write the filter parameter to the last name attribute on the **Main** entity.

   c. In the **Required** column, confirm that **Required for LastName** is selected. You need to enter a last name to proceed.

   d. Click **Next**.
5. Choose the parameters to be returned as output. The data to be returned is in the **AccountList** recordset on the **NameSearchResults** entity.

   In the Output Parameters dialog box, click in each **Read parameter from** cell and select the following data sources from the drop-down menu:

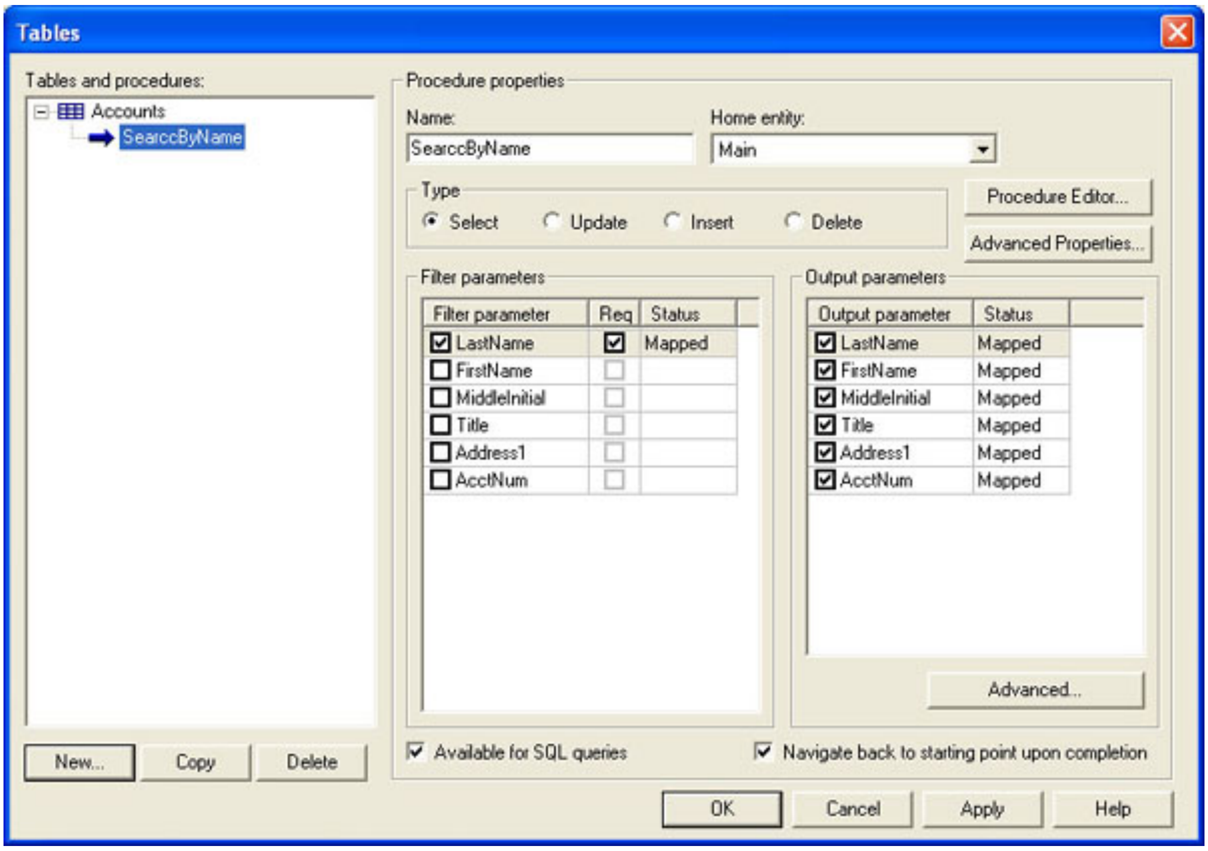
Parameter	Data Source
LastName	NameSearchResults.AccountList.LastName
MiddleInitial	NameSearchResults.AccountList.MiddleInitial
Address1	NameSearchResults.AccountList.Address1
AcctNum	NameSearchResults.AccountList.AcctNum

   This maps the procedure fields with the data in the **AccountList** recordset on the **NameSearchResults** entity.
6. Click **Next**.
7. View the Summary, and click **Finish** to return to the Tables dialog box.

Review the information in the Tables dialog box for the `SearchByName` procedure. You can expand **Accounts** to view the procedure.

Verify that the **Home** entity is **Main**. This table begins and ends at the **Home** entity. Each procedure must have a home entity to ensure that queries and procedures begin from a known point in the host application.

10. Click **OK** and save your model.



Name: SearccByName

Home entity: Main

Type: [✓ Select] Update Insert Delete

Filter parameters

Filter parameter	Req	Status
☑ LastName	☑	Mapped
☐ FirstName	☐	
☐ MiddleInitial	☐	
☐ Title	☐	
☐ Address1	☐	
☐ AcctNum	☐	

Output parameters

Output parameter	Status
☑ LastName	Mapped
☑ FirstName	Mapped
☑ MiddleInitial	Mapped
☑ Title	Mapped
☑ Address1	Mapped
☑ AcctNum	Mapped

☑ Available for SQL queries

☑ Navigate back to starting point upon completion

Figure 13--SearchByName procedure details

Test and Deploy the Model

There are multiple utilities you can use to test your model before you deploy it. In this tutorial we are going to use two debug utilities, Procedure Test and SQL Test.

Because the model you created is a simple, two-entity model, comparing the entities and identifying the traversal path is easy. But consider a model that contains hundreds of entities with a multitude of attributes and complicated navigation, and you can understand how all of these testing utilities become important debugging tools:

Validator checks a model's completeness.

Signature Analyzer ensures each screen is uniquely identified.

Navigator reviews the model's traversal path.

Procedure Test confirms procedures run correctly.

Model Debug Messages diagnoses problems such as synchronization with the host by showing the model's behavior while interacting with a terminal datastream.

SQL Test verifies that SQL results are correct.

To run a procedure test

From the **Debug** menu, click **Procedure Test**. Since you have only one table and one procedure, they are selected by default.

In the **Procedure Filters** box, type `W` (uppercase W) in the **Value** field (associated with the **LastName** filter.)

Click **Execute**. Seven entries should be returned and displayed in the **Procedure Outputs** box.

Click **Clear** and enter `K` (uppercase K) to test your procedure again.

Click **Execute**. You should see one entry under Procedure outputs.

Click **Close**.

To run an SQL test

From the **Debug** menu, click **SQL Test**.

2. In the SQL statement box type the following SQL command:

   `SELECT * FROM accounts WHERE lastname LIKE W`

3. Click **Resolve** to resolve the SQL statement to a procedure.
4. Click **Execute**.

   Seven records are returned and displayed in the **Output recordset** box.

5. Click **Close** and save the model.

The Host Integrator supports a subset of the SQL 92 standard for SELECT, UPDATE, INSERT, and DELETE statements. Some of these commands are used in a slightly different manner. For more details, see SQLSyntax.

Testing Web services

In Host Integrator, Web services are automatically provided by the session server as an embedded SOAP stack after a model package is deployed using the Design Tool. As the provider of a Web service, you publish the WSDL document to give developers access to your Web service. With the WSDL-generation URL, developers can use utilities to generate specific files to locate and consume the Web service.

This location lists the available Web services WSDL documents:

`http://<session server>:9680/vhi-ws`

Optionally, you can access the WSDL for a specific model by using the model name. For example:

`http://<session server>:9680/vhi-ws/model/<model name>?wsdl`.

Testing your service is an important step before you run your process in a production environment. Testing is an easy two-step procedure:

After you deploy the model to the Host Integrator Server, a message box informs you that the model is successfully deployed.

Figure 14 -- Deployment successful message

1. Click **Test** to launch the Web Services Explorer, a browser-based test tool, where you can test the Web service.
2. In the **Actions** view of Web Services Explorer, enter the parameters of the **WSDL** in order to test the service and click **Go**.

> **Tip**
>
> To test Web services that are protected by WS-Security, Basic Authentication, or HTTPS, use a third-party development environment, such as Microsoft Visual Studio.

Although the way that Web services are implemented in Host Integrator is beyond the scope of this tutorial, you can read more about them in About Verastream Web Services.

Extending Your Model with Event Handlers

Event handling extends the capabilities of Host Integrator models by interrupting the interpretation of a model and turning control over to user-supplied procedural code. For example, you can use event handlers to:

- Convert cryptic host application codes to user-friendly descriptions
- Convert formats for currencies or dates
- Secure access to sensitive data
- Create other functions that might otherwise require custom client-side programming

You can also use event handlers to increase Host Integrator error-handling capabilities. For example, you can add an event handler at the point an error occurs and then implement event-handler code that intercepts the error, takes control, and corrects the error.

Although event handlers are beyond the scope of this tutorial, you can read more about this powerful feature in About Event Handlers.

About Deploying Host Application Models

To move into production testing, you need to deploy your model to the Host Integrator server that handles the communication between your custom application and the model, and between the model and the host application. When deploying models, use either:

Design Tool Deployment - If you are deploying one model to one server (local or remote), with one model configuration, use the deployment options on the Design Tool's File menu.

Command Line or Script Deployment - If you are deploying a model to one or more production servers, use deployment commands from a command line or in a script. This enables you to automate the model's deployment to each Host Integrator server.

For more information about using these different deployment methods, see Deploying Models.

Deploying MyModel.modelx

You are using your workstation as the development, testing, and production environments; therefore, you will deploy your model to the local session server, localhost:

Open your model in the Design Tool.

To deploy your model, click **File > Deploy to Local Server**.

When the dialog box informs you that you have successfully deployed your model, click **OK**.

Close the Design Tool.

3.4.5 Creating a Web Builder Project and Generating a Web application

After you have completed, tested, and deployed the model, you are ready to create a project from the Host Integrator model. For the CICSAccts example, you will use Web Builder to generate Web pages from the model.

Web Builder generates projects ranging from simple screen-based rejuvenation to full procedure-based integration. A rejuvenation Web application uses the model's entities, attributes, and recordsets as the basis for creating the Web front end. An integration project uses procedures to create a Web application or component interfaces, JavaBeans, and .NET class libraries.

You can use Web Builder to create an HTML 5 Web application that can be used either *out of the box* or customized by your Web application developer. Screen-based rejuvenation may be adequate for some projects, but, in this tutorial, you will use the model's procedures to integrate the CICSAccts application into a more sophisticated application.

> **Note**
>
> The steps you follow to generate a project are the same on either a .NET or Java platform. Select the type of project you want to build based on your platform and whether you want to generate a Web application or component interface.

To build an HTML 5 Web application For CICSAccts

Open the Design tool and then choose **File** > **Web Builder**.

In Web Builder, click **New** to create a new project.

Select **HTML 5 Web Application** as the type of project to create.

Select **MyModel** as the Host Integrator model name.

Enter `MyModel_Application` as the project name.

6. Click **Properties** and then **Build**.

   When the Build Project dialog box displays a `BUILD COMPLETED` message, your Web application automatically runs. (You can run the project manually in the Web Builder by selecting your project and clicking **Run**.)

7. When the Web application displays, click to open the available Procedures.

8. Click **Search by name**, enter a `W` in the **LastName** text box, and then click **Execute**.

   Close the application and Web Builder.

Deploying the Project to the Enterprise

In a work environment, after you have developed the new Web or client/server application and the application is tested, you can deploy it to the enterprise using the customer's preferred hardware platform and network configuration.

For specific details on deploying a project, review:

- Deploying Models
- Deploying Web Services

3.4.6 Congratulations!

Using Verastream Host Integrator, you have created an easy-to-use Web application that accesses the CICSAccts host application. You have:

- **Defined the business needs for the project.**

  You need a clear understanding of the business problem to be solved by your solution.
- **Developed the model requirements and mapped the data.**

  In the "real" world, you would work with system administrators to understand the user's computing environment, programmers to learn what programming languages are preferred, and host application users to understand how the host application functions. This would lead to setting the model requirements and mapping the host application data. During this exercise, you were given the model requirements and mapping information.
- **Built and deployed the model.**

  Using the Design Tool, you created a model of the host application, which was automatically deployed to your local Host Integrator Server along with the automatically generated Web service. You have:

  Created patterns to uniquely identify each entity

  Identified operations to navigate through the host application

  Configured attributes for data entry and retrieval

  Used recordsets to handle scrolling data

  Created tables of host data

  Developed procedures to query and retrieve data from those tables
- **Created a Web project using Web Builder.**

  You generated a Web application to test the procedures in your model, and delivered this Web application as a simple front-end to the host application. You also have access to a Web service to deliver to Web programmers for a more complex application that integrates host data through the model.
- **Deployed the project to the enterprise.**

  Because you did not move the CICSAccts application into production, you didn't have to perform this step.

Although your project will be more complex than this straightforward example, the steps you perform and the utilities you use are the same each time you create, deploy, and implement a Verastream Host Integrator model.

4. Using the Design Tool

4.1 The Modeling Process: Getting Started

To begin the modeling process:

Open the Design Tool.

On the File menu, click New to specify all the necessary settings in the New Model dialog box. When you have finished supplying the necessary settings, provide a model name, and click OK.

You should now be connected to your specified host.

Now, you're ready to add an entity to your model.

4.1.1 Creating a Model

With the Design Tool, you create a model of the host application. The model consists of a main model file and several supporting files that are located in the `<documents>\RocketSoftware\Verastream\HostIntegrator\models\<your model folder>`. The supporting files saved in your models directory vary depending on whether you are saving your model has a .model file or as a .modelx file.

For more information:

- Adding entities
- Adding patterns
- Adding attributes
- Adding recordsets
- Adding operations
- Working collaboratively
- Creating tables
- Creating procedures
- Using event handlers
- Deploying models

You can use Web Builder to quickly and easily generate a web application or component interface such as a web service or JavaBeans, based on the table of a host application model, or based on a host application screen layout.

More information

- Learning to Use Host Integrator
- Host Integrator development process

4.1.2 Configuring a Model

The Design Tool provides several ways to configure unique properties and settings that enhance the capabilities of the host application model. You can use the pre-configured settings files provided with Host Integrator to start with the defaults appropriate for the terminal session you are modeling. Settings include:

- Variables - which allow attributes to be accessible from all parts of the Host Integrator
- Model preferences - to customize default settings for the Design Tool
- Model properties - to configure settings for the current model
- Advanced entity properties - to configure settings for the current entity
- Character mode - to configure settings for character mode terminal types (for example, VT)
- Events- to configure synchronization for character mode terminal types
- Command lists - to create a login, a logout, or a move cursor command list
- Descriptions - to add descriptions in exported documentation created for the model.

More information

- Model examples

4.2 Work Collaboratively

When you are building your host application project files it is often important for multiple people to contribute to a project. Configuration management software, such as Git, provides your team with the ability to manage and merge your project files efficiently. Because Host Integrator saves your project files in an XML format, making them accessible to multiple developers, it is easy to work collaboratively.

Each host application project contains a number of files that you need to keep together in order to ensure the project's uniqueness. Files are generated and saved in a directory that uses the same name as the model you are using. The main model file, .modelx, uses that same name. It is important to keep all the project files together and preserve their folder structure.

The `<Model name>` directory, which, by default, is created in my documents\RocketSoftware\Verastream\HostIntegrator\model. Each Model name directory contains the following files and directories:

- The `<model name>`.modelx and modelx_1.xsd files.
- The `<entity name>`.entityx and entityx_1.xsd files, in the entities subfolder.
- The `<table name>`.tablex and the tablex_1.xsd files, in the tables subfolder.
- The Scripts directory.

In particular, build.xml, the `\src` subfolder, and the `\lib` subfolder are required for event handler maintenance.

In previous versions of Host Integrator, project files were saved with a .model extension. By default your project is saved using its existing extension. Using the Save As option you can choose which format to use. If you choose to save your file as a .model file, the directory contains these files:

- The `<model name>`.model file.
- The .snapshot file.
- The Scripts directory.

In particular, build.xml, the `\src` subfolder, and the `\lib` subfolder are required for event handler maintenance.

4.2.1 Using source control to manage and merge changes

Using the multiple plain text files generated in the Design Tool and a tool, such as Git, different developers can work on the different files needed to build your host application project.

4.2.2 Things to remember

- It is always better to make changes to the model using the Design Tool. In the Design Tool all objects are available to copy from the user interface and every change is validated to make sure that the model stays consistent.
- Because the objects are available in XML format, you can open the modelx, entityx and tablex files in an editor. Make sure that the editor you choose is capable of using the associated .xsd files. The .xsd file checks the syntax and possible element values, but it is up to the developer to check the valid syntax before attempting to run the project.
- If your projects are referencing the same mainframe application, you can copy and rename the entire .entityx or .tablex file into the destination model. You can copy entityx or tablex files from one

model directory into another model directory. If the name in the destination model already exists, you can simply give the new entityx or tablex files a new name. The new names will be available as entities or tables in the Design Tool.

- One advantage of an editor is the ability to globally copy and rename an object. You can copy information within the same modelx file to create a number of variables that use the same properties or within a file (modelx, entityx, or tablex) of one project, you can copy a complete definition of an object, such as a variable, or copy a part of an object such as a pattern or a recordset, belonging to an entity, or copy one or more columns that are part of a table.

  For example, it is easy to copy the information contained within the variable tags, and then add this variable to another project by pasting it after an existing variable in your XML file. You must rename the variable to an unique value:

```
    <Variable>
    <Name>newUserID</Name>
    <InitOption>VarInitUninitialized</InitOption>
    <VarKind>VarKindSetting</VarKind>
    <Attributes>
      <VarAttrsRead/>
      <VarAttrsWrite/>
    </Attributes>
    <Setting>HostUserName>/Setting>
    </Variable>
```

4.2.3 Importing model elements

With this option, developers can work separately on different sections of the same model, import sections into the destination model, and reuse model elements as often as needed. You can import elements from older formatted models into the current format and vice versa. You are not restricted by the different versions of existing project files. There is complete documentation on this option available. See Importing Model Elements.

4.2.4 Optional Files

The `Recordings` directory, which is where the Host Emulator trace files are stored is optional. If you do not intend to load and run a model recording in the Host Emulator, this directory is not needed.

Note

If you want to save the settings for this model so that it will be the basis for creating other models, select Save Settings As. The `.dtool` file you create can include Design Tool-specific configuration information such as window size, colors, keymapping, and preferences, as well as information about the host application and connection settings.

More information

Validating host application models

Troubleshooting tips and techniques

4.3 Setting Up a Connection

4.3.1 Design Tool Connection Settings

Entity Window Options

Terminal Window Menus

Entity Window Options

The Entity window contains all of the settings used to define the host screens that make up the model. The Entity box contains all of the currently defined entity names with visual indicators to indicate whether each entity listed is "reachable" from the current location via dynamic traversal or navigation.

When you are in offline mode, all screens are considered reachable and display a green icon since the screen shots are loaded from a .snapshot file which has also recorded the navigation of the model. Next to the Entity box, there are three buttons that allow a user to add a new entity, delete the current one, or view the Advanced Entity Properties dialog box.

There are five tabs available from the Entity panel:

Pattern tab - contains all of the settings used to define patterns on a selected entity.

Attribute tab — contains all of the settings used to define attributes on a selected entity.

Operation tab - use this tab to edit or define operations on a selected entity.

Recordset tab - contains all of the settings used to define recordsets on a selected entity.

Cursor tab - contains cursor movement settings to be used with character-mode hosts. (If you are not working with a character-mode host, Cursor tab controls are unavailable.)

See Entity Settings for detailed information.

Terminal Window Menus

The Terminal window contains all of the menu items that can be used to configure host screens that make up the model.

The following menus are available on the Terminal window:

File

Edit

Connection

Settings

Events

Model

Debug

Window

Help

See Terminal Settings for detailed information.

4.3.2 Entity Settings

The Entity window contains all of the settings used to define the host screens that make up the model. The Entity box contains all of the currently defined entity names with visual indicators to indicate whether each entity listed is "reachable" from the current location via dynamic traversal or navigation.

When you are in offline mode, all screens are considered reachable and display a green icon since the screen shots are loaded from a .snapshot file which has also recorded the navigation of the model. Next to the Entity box, there are three buttons that allow a user to add a new entity, delete the current one, or view the Advanced Entity Properties dialog box.

There are five tabs available from the Entity panel:

Pattern

Attribute

- Operation
- Recordset
- Cursor

See Adding Entities to a Model for information on how to add and define your entity.

4.3.3 Using SSH: Overview

You can configure SSH connections when you need secure, encrypted communications between a trusted host and your PC over an insecure network. SSH connections ensure that both the client user and the host computer are authenticated; and that all data is encrypted. Passwords are never sent over the network in a clear text format as they are when you use other protocols, such as Telnet.

- Data Encryption Standards
- Data Integrity
- Digital Signatures
- How does SSH Work?
- SSH Authentication Options
- Using Model Variables for SSH Authentication
- Public Key Authentication
- Enter Username and Password

Data Encryption Standards

Encryption protects the confidentiality of data in transit. This protection is accomplished by encrypting the data before it is sent using a secret key and cipher. The received data must be decrypted using the same key and cipher. The cipher used for a given session is the cipher highest in the client's order of preference that is also supported by the server. You can use the cipher list on the Advanced VT SSH dialog box to specify which ciphers the SSH connection should use.

Verastream Host Integrator supports the following data encryption standards:

- AES (also known as Rijndael) (128-, 192-,or 256-bit) CBC mode and CTR mode
- TripleDES (168-bit) CBC mode
- Blowfish CBC
- Cast 128
- arcfour 128
- arcfour

Data Integrity

Data integrity ensures that data is not altered in transit. SSH connections use MACs (message authentication codes) to ensure data integrity. The client and server independently compute a hash for each packet of transferred data. If the message has changed in transit, the hash values are different and the packet is rejected. The MAC used for a given session is the MAC highest in the client's order of preference that is also supported by the server.

Verastream Host Integrator supports the following MAC standards:

SHA256

SHA1

SHA512

MD5

RIPEMD160

RIPEMD160 openssh.com

SHA1-96

MD5-96

If your SSH server on the host supports it, you always have the option of selecting **None** when choosing a MAC or data encryption standard.

Note

The values for both the MAC and data encryption standards that you can select are dependent on whether the **Only show FIPS validated values** is enabled. This option filters the values available.

Digital Signatures

Digital signatures (user key and host key algorithms) are used for public key authentication. The authenticating party uses the digital signature to confirm that the party being authenticated holds the correct private key. The SSH client uses a digital signature to authenticate the host. The SSH server uses a digital signature to authenticate the client when public key authentication is configured.

Verastream Host Integrator supports the following digital signature algorithms:

ECDSA

RSA

DSS

EdDSA

The known_hosts file

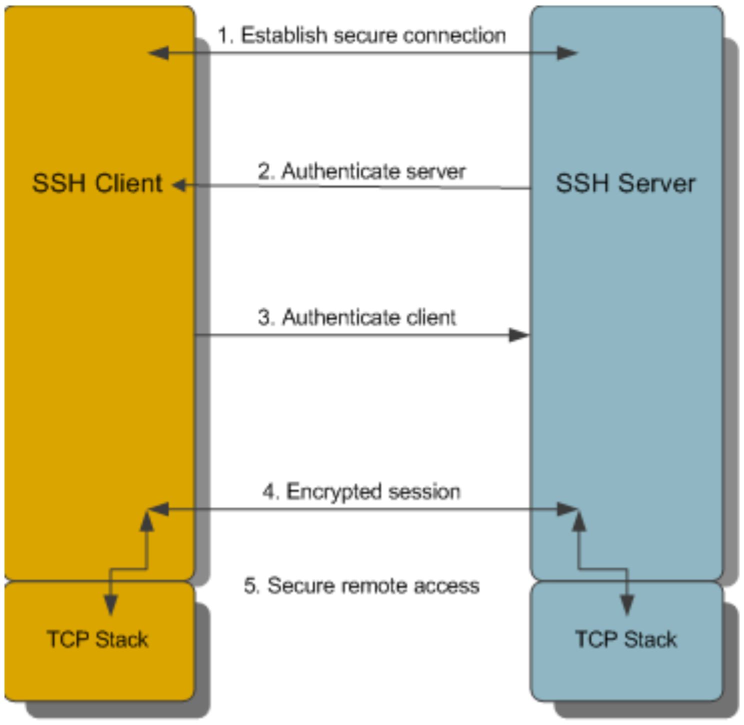
Every time an SSH client connects to a host, it stores a host key for that host. These stored host keys are referred to as known host keys or just known hosts. In OpenSSH, these known hosts are stored in `/etc/ssh/known_hosts` and in `.ssh/known_hosts` in each user's home directory. VHI uses the known_hosts file to verify the identity of the server the model is going to connect to.

You can add the remote host's public key to the user's known_hosts file using either SSH or VHI.

By default, VHI uses the known_hosts file, located in directory `~/.ssh`. You can specify a different file by using predefined model variable `KnownHosts`. See Using Model Variables for SSH Authentication.

How does SSH work?

These are the basic steps involved in creating a SSH channel to transmit data securely. It is assumed that the SSH Server is trusted and present in the known_hosts file.



1. Establish a secure connection

   The client and server negotiate to establish a shared key and cipher to use for session encryption, and a hash to use for data integrity checking.

2. Authenticate the server

   Server authentication enables the client to confirm the identity of the server. The server has only one chance to authenticate to the client during the authentication process. If this authentication fails, the connection fails.

3. Authenticate the client

Client authentication enables the server to confirm the identity of the client user. By default, the client is allowed multiple authentication attempts. The server and client negotiate to agree on one or more authentication methods.

4. Send data through encryption session

   Once the encrypted session is established, all data exchanged between the SSH server and client is encrypted.

5. A channel is created and a terminal emulation using the terminal type specified in the configuration dialog box is started

   Users now have secure remote access to the server and can execute commands through the secure channel.

More information

Configuring a VT session

Using Model Variables for SSH Authentication

Advanced VT SSH Options

SSH Authentication Options

You can enable authentication in two ways; interactively using the Design Tool or using model variables in the Session Server and Design Tool.

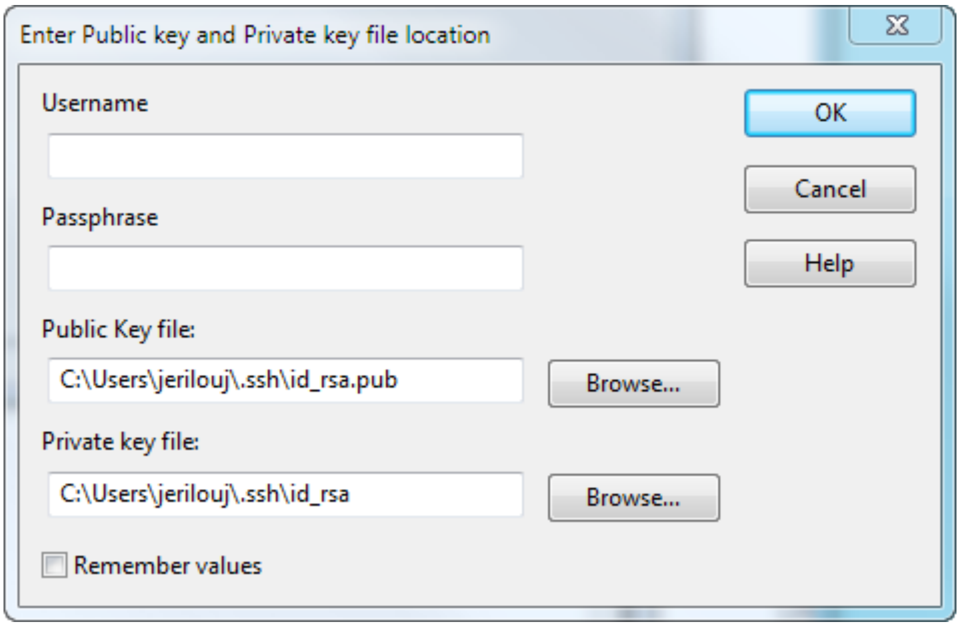
To authenticate using model variables, see Using Model Variables for SSH Authentication. This is the preferred option.

If you have not configured authentication using model variables in the Design Tool, you are prompted to specify the authentication values:

- Password

  Specify the login username and password for that user on the SSH server host. The password is sent to the host through the encrypted channel.

- Public key and Private key



Username

Passphrase

Public Key file: C:\Users\jerilouj\.ssh\id_rsa.pub

Private key file: C:\Users\jerilouj\.ssh\id_rsa

Remember values

Specify the username, passphrase, and location of the public and private key files.

Relies upon public/private key pairs. Public keys and private keys are pairs of cryptographic keys that are used to encrypt or decrypt data. Data encrypted with the public key can only be decrypted with the private key; and data encrypted with the private key can only be decrypted with the public key.

To configure public key authentication, each client user needs to create a key pair and upload the public key to the server. If the key is protected by a passphrase, the client user is prompted to enter that passphrase to complete the connection using public key authentication. Public keys are not sensitive information and may be known to anybody, whereas the private key is protected very carefully by a strong passphrase.

Caution

If you are using utility ssh-keygen to create the private and public key for SSH, be aware that the newer ssh-keygen versions default to an OpenSSH format to generate private keys. This is not supported by VHI. The public/private key pair must be in PEM format. Verify that the header of the private key contains the text, RSA PRIVATE KEY. You can convert keys with OpenSSH private key format using ssh-keygen to the old PEM format. Use the command `ssh-keygen -m PEM -t rsa` to generate the files `id_rsa` and `id_rsa_pub` in the correct format.

- SSH Agent

  Specify the username.

  SSH agent is a program to hold private keys used for public key authentication (ECDSA, RSA, DSA). Host Integrator connects to the agent for authentication.

The SSH agent:

Stores keys securely in encrypted form

Enables you to only specify a username when connecting using SSH. Host Integrator connects to the SSH agent, and the agent takes care of the needed authentication. You do not have to specify a password, key, or passphrase.

If you plan to authenticate using public keys, before you configure Host Integrator:

Verify that the SSH agent you are using is available and configured on either your Windows or Linux system

Start the SSH agent and specify the location of the private key file. When the keys require a passphrase that should be entered as well. The agent is now running as a daemon at the background.

Note

The key data must be in OpenSSH format. Remove any new lines, comments, or other data. Whatever tool you use to create the private key, must be used to export the key to OpenSSH format. If the public key is in SSH2 (SECSH) format, run the following OpenSSH command to convert the certificate from SSH2 to OpenSSH: `ssh-keygen -i -f ~/.ssh/id_dsa.ssh2.pub > ~/.ssh/id_dsa.pub`

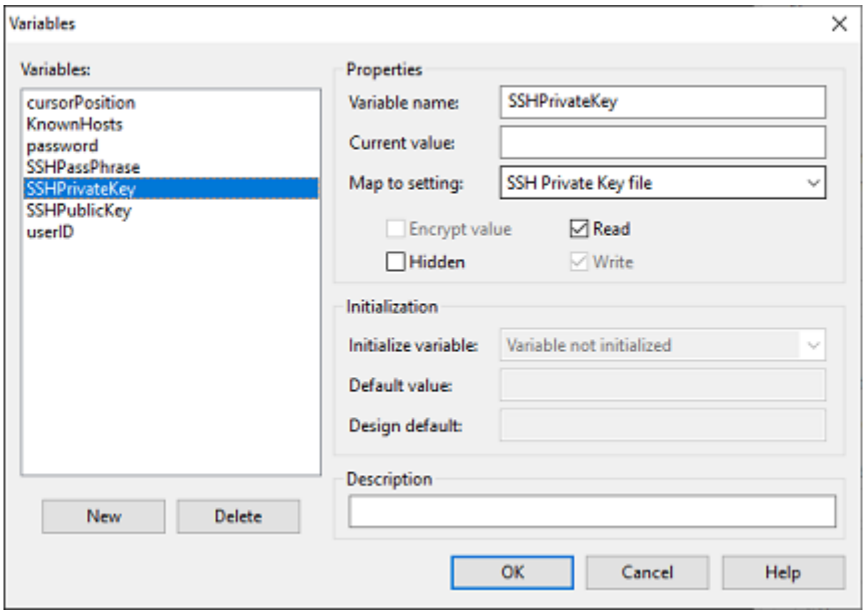
On each of the authentication dialog boxes you can enable the option Remember values. When selected, the values you entered are assigned to the associated model variables.

Using Model Variables for SSH Authentication

You can specify authentication credentials using model variables. Model variables are placeholders for data and are very useful when specifying fixed values, such as host user IDs and passwords.

A list of default variables are listed when you create a new model. If you selected **Remember values** when setting values using the Design Tool, the values are assigned to the associated model variable.

Special character `~` can be used to specify the home directory, independent of the platform. For example, `~/.ssh/id_rsa.pub` refers to the public key in the users `.ssh` home subdirectory.


Variables

Variables:

cursorPosition
KnownHosts
password
SSHPassPhrase
SSHPrivateKey
SSHPublicKey
userID

Properties

Variable name: SSHPrivateKey

Current value:

Map to setting: SSH Private Key file

☐ Encrypt value ☑ Read

☐ Hidden ☑ Write

Initialization

Initialize variable: Variable not initialized

Default value:

Design default:

Description

Variable name	Map to setting
userID	Host User Name
password	Host password
SSHPublicKey	SSH Public key file
SSHPrivateKey	SSH Private key file
SSHPassPhrase	SSH passphrase for the private key
SSHKnownHosts	File with public keys of SSH servers to connect to

Setting model variable values in the Administrative Console

Model variable values that are set using the Design Tool are stored in the model file. To remove this sensitive data from the model file and store it on the Session server:

After setting authentication variable values in the Variables dialog box, test the model.

If successful, remove the values for the password, username, and passphrase variables.

Deploy the model to the Session server.

In the Administrative Console, open the Properties page for the session pool associated with the model and re-enter the model variable values. If you need to provide a unique model variable value for each session in the pool, create a model variable list that contains a set of values for the model variables in the model on which your session pool is based.

If the Session server is not able to connect because the values are not set, an error message is written to the Session server log file: `An error occurred in communications - SSH password authorization failed (username/password).`

Public Key Authentication

Enter your VT host credentials to log on to the host. This dialog box appears if you are using SSH public key authentication to connect.

- Username - Enter your VT host username to log on to the host.
- Passphrase - Enter the passphrase to decrypt your private key. If your private key is not encrypted, leave this field blank.
- Public Key file - Enter or browse to the location of your public key file.
- Private Key file - Enter or browse to the location of your private key file.

Enter Username and Password

You must supply authentication credentials to establish a connection to the host.

Supply a username to log on to the host.

For the connection to be established, supply the needed password.

This dialog box appears if you have not specified values for the model variables.

4.3.4 Using SSL/TLS

Telnet Secure Socket Layer (SSL) and Transport Layer Security (TLS) security protocols are available for 3270 and 5250 session types, and Telnet Extended SSL/TLS support is available for 3270 session types. These Telnet options apply to the connection between a host and the Host Integrator session server or Design Tool. They do not apply to the connection between the client and the Host Integrator session server.

How to enable TLS/SSL encryption

To configure SSL/TLS encryption in your model:

1. The Design Tool must be offline and disconnected.
2. To modify an existing model, click Connection > Session Setup. To configure a new model, click File > New to display the New Model dialog.
3. Select the Transport "Use SSL/TLS" checkbox (for 3270 or 5250).

4. After connecting to the host using the Design Tool, you can determine the negotiated cipher, see Settings > View Settings > Host Communication > Telnet > Secure Host SSL Negotiated Cipher. The TLS protocol version and negotiated cipher are also logged in model debug messages (.vmr files).

Enabling FIPS 140-2 Validated Encryption

The Federal Information Processing Standards (FIPS) is used by US government agencies. When using TLS/SSL, you can enable FIPS 140-2 validated encryption. To enable this feature, set an operating system environment variable, `VHI_FIPS=1`, before you start the session server or Design Tool.

Note

On Linux, you may need to export the environment variable so it is available to the process running the session server.

To confirm FIPS 140-2 encryption is enabled:

1. Open the Administrative Console.
2. You can verify that FIPS is enabled on the Session Server > Properties > General > Security panel and in the session server log. FIPS mode is not supported on the IBM AIX platform.

Certificate Checking

This section describes how the Design tool and Session Server verify a secure connection to a 3270 or 5250 host. It does not apply to other TLS connections made by, or to, VHI components.

As part of the standard TLS handshake, the host will send a chain of certificates to the Design tool or Session Server.

The following checks are performed: 1. The certificate chain from the host must be complete, and signed by one of the trust anchors in the machine's trust store. 2. The name of the host must be equal to one of the **Subject Alternative Names** (SAN) listed in the server certificate. Note that in VHI, the host name can be specified in the model or in the deployment descriptors.

VHI does not perform revocation checks.

VHI does not download CA certificates using AIA extensions.

If the host presents a certificate chain from a well-known CA, then your machine's trust store should already be correct, and no changes are needed. If the host presents a self-signed certificate, or if the administrator has used a proprietary CA, then you can modify your trust store. Do not grab certificates from the connection, always contact the administrator of the host and ask for the correct certificate(s) to import.

Note

The trust store used is not part of VHI, it is global to the machine on which the Design tool or Session Server are running. If you modify this trust store, these changes may affect other programs. It also means that uninstalling VHI will not revert those changes.

Importing certificates

Windows: you can import a certificate using `CertMgr /add <filename> /s /r localMachine root`.

Alternatively, you can use `certlm.msc`. Place the certificate into the "Trusted Root Certification Authorities".

By importing the certificate into the "localMachine" store it becomes available to all users, which is necessary for the Session Server.

Linux: you can import a certificate (in PEM format) using `sudo trust anchor <filename>`.

Validation Errors

Error 3055 indicates a validation error and contains a detail string. Here are a few validation errors that can occur.

unable to get issuer certificate
The issuer certificate of a locally looked up certificate could not be found. This normally means the list of trusted certificates is not complete.
Solution: ask the administrator to configure the host to send all intermediary certificates in the TLS handshake.
Solution: contact the administrator of the host and ask for the correct certificate(s) to import.

certificate has expired
The certificate has expired: the notAfter date is before the current time.
Solution: ask the administrator of the host to install a new certificate.

self-signed certificate
The host certificate is self-signed and the same certificate cannot be found in the list of trusted certificates.
Solution: import the self-signed certificate into the trust store.

self-signed certificate in certificate chain
The certificate chain could be built up, but no suitable trust anchor (which typically is a self-signed root certificate) could be found in the trust store.
Solution: contact the administrator of the host and ask for the correct certificate(s) to import.

host name mismatch
The name of the host in the model or in the model's deployment descriptor does not match.

Solution: change the model or the deployment descriptor to use one of the SANs in the host certificate.

ip address mismatch
You are connecting to the host using an IP address and it is not one of the SANs.
Solution: connect using a name, not an IP address. Certificates with a SAN IP address are rare.

Other errors can also occur. Refer to the OpenSSL documentation for a full list.

Altering the configuration of TLS connections

The configuration of host TLS connections is performed using an OpenSSL Configuration file, `%VHI_ROOT%/bin/openssl-vhi.cnf`. In this file, you will find a section `[hostssl]` where you can make changes to the configuration.

Note that the configuration file is read once, at startup. To see the effect of your edits, restart the Design Tool or Session Server.

Disabling TLS 1.3

If your host fails to negotiate TLS 1.3 connections, you can disable the use of this protocol version. In the configuration file, change `MaxVersion = TLSv1.3` to `MaxVersion = TLSv1.2`.

Enabling SSL 3.0, TLS 1.0, or TLS 1.1

TLS 1.1 and earlier protocol versions are disabled by default. If your host does not yet support TLS 1.2 or TLS 1.3, you may see errors related to TLS version not supported in Design Tool, the session server log, or model debug messages (.vmr file). To enable these protocols:

change `MinVersion = TLSv1.2` to `MinVersion = TLSv1.1`, `MinVersion = TLSv1`, or `MinVersion = SSLv3`

uncomment the `CipherString` line by removing the initial `#` character

In this line, change `:@SECLEVEL=1` to `:@SECLEVEL=0`.

Modifying the ciphers offered to the host

If you wish, you can control the ciphers offered in the handshake. The `CipherString` setting controls the ciphers used for TLS 1.2 and earlier; the `Ciphersuites` setting controls the ciphers used for TLS 1.3. Refer to the OpenSSL 3.0 documentation to see possible values for these settings.

Client Authentication

If the host requires client authentication from Host Integrator, your private key and client certificate chain must be stored in a PEM file.

If your file is named `%VHI_ROOT%/securehost/certificate.pem` add the following line to the hostssl section:

`Certificate = ${ENV::VHI_ROOT}securehost/certificate.pem`

The file must be in PEM format with the unencrypted private key and the certificate chain in chain order.

If your certificate and private key are in PFX format, you can convert it to OpenSSL PEM format using the OpenSSL command line utility in the `%VHIROOT%/bin` folder.

Changing the Linux trust store

The actual location of the trust store on Linux is not standard. On startup, the code will test for the presence of an ordered list of candidates. If a file is present, it will be used as the trust store.

`/etc/ssl/certs/ca-certificates.crt`

`/etc/pki/tls/certs/ca-bundle.crt`

`/etc/ssl/ca-bundle.pem`

`/etc/pki/tls/cacert.pem`

`/etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem`

`/etc/ssl/cert.pem`

You can change the file used as the trust store by changing the `VerifyCAfile` setting.

Disabling Certificate Validation

For troubleshooting purposes, you can temporarily disable verification. In the hostssl section, remove the line `VerifyMode = Peer` to disable verification.

Permanently disabling verification poses a security risk and is not recommended.

4.4 Configuring Sessions

4.4.1 Configuring a Host Session

Use the Session Setup or the New Model dialog box to connect to a host and specify emulation options. To open the Session Setup dialog box, click Session Setup on the Connection menu.

Note

You can open the Session Setup dialog box if you have not connected it to a host session. Otherwise, this option is unavailable. Use the Connection Properties dialog box to review settings after you are connected.

The options in the Session Setup dialog box vary according to the values you select for session type and transport type. Click one of the following for more information:

- Configuring a 3270 terminal session
- Configuring a 5250 terminal session
- Configuring a VT terminal session
- Configuring an HP terminal session

Once you have set up the options in this dialog box, you can click Connect to connect immediately, or click OK to close the dialog box. If you click OK, you can always connect later. Connection settings can be saved to a settings file (.dtool) with other configuration information.

Configuring a 3270 Terminal Session

The following options are available in the Session Setup or New Model dialog box.

After you connect, you can review these settings in the Connection Properties dialog box.

Options	Description
Session Type	Specifies the type of session to configure.
Model ID	Specifies the terminal (also known as a display station) you want the Design Tool to emulate.
Transport Type	This setting specifies the transport type being used to connect to the host. You can use Telnet, Telnet Extended, SSH, or NS/VT transport types as well as Secure Socket Layer (SSL) and Transport Layer Security (TLS) security protocols. Telnet is available for 3270,5250, VT, and HP session types, while Telnet Extended is only available for IBM 3270 session types. Choose SSH to connect to a VT host using SSH security protocols. NS/VT is available for HP session types.
Use SSL/TLS	See Using SSL/TLS
Host Name or IP Address	Use this box to identify the host to which you want to connect. You can either select a host from the drop down list or type one into the host name field. When you start the Design Tool without a value for this setting, the Design Tool looks for the Hosts file in the same folder as Wsock32.dll. If you are using third-party TCP/IP software, your Hosts file may be in another location. When it finds a Hosts file, the Design Tool changes the value of this setting.

Options	Description
Port Number	Specifies the host port or socket number that the Telnet session should use. You can enter any number between 0 and 65,535 in this field. You can configure Port in either the Session Setup or View Settings dialog boxes. The default port number for Telnet is **23**.
Device Name	Specifies the Device name (also known as LU name) to which to connect. Supports up to 32 characters. You can map the Device Name setting to a model variable in the Model menu > Variables. This allows a different setting for each server runtime session. If a model variable is mapped to the Device Name setting, the Device Name configured in Session Setup will be used as a default unless changed by a session pool model variable list or a Host Integrator API SetModelVariable method.

More information

- Using SSL/TLS
- Using SSH

Advanced 3270 Telnet and Telnet Extended dialog boxes

You can configure the following options in these dialog boxes:

Option	Description
Terminal ID	In a 3270 session with a Telnet transport type, this setting overrides the Model ID selected in Session Setup. It allows you to specify a Terminal Model that Host Integrator does not implicitly support (but might work with), such as Fujitsu. For best results, specify one of the Terminal Models available in Session Setup and leave the Terminal ID box empty. When the Terminal ID box is empty, the Design Tool uses the Model ID defined in Session Setup by default. Only specify a Telnet Terminal ID, if after experimenting with other available Terminal Models IDs in Session Setup, you still cannot connect to the host. If you must define a Telnet Terminal ID, specify a string you know is acceptable to the host. Otherwise, you may experience problems connecting to the host and emulation problems after connecting to the host. Typically, these types of problems occur if the host is not configured to recognize the terminal specified in the Telnet Terminal ID string.

Option	Description
Telnet Location	Specifies where the connection originated. Can also be used to provide informational messages to the host from the computer. You are not required to enter anything in this box. Supports up to 260 characters. You cannot change this value while you're connected to a gateway.
Send Keep Alive packets	To become aware of connection problems as they occur, you can configure your model to Send Keep Alive packets.
Keep Alive Timeout	The interval between the keep alive requests sent by the Design Tool. The range of values is 1-9999 seconds, and the default is **600** seconds.

Configuring a 5250 session

The following options are available in the Session Setup or New Model dialog box.

After you connect, you can review these settings in the Connection Properties dialog box.

Options	Description
Session Type	Specifies the type of session to configure.
Model ID	Specifies the terminal (also known as a display station) you want the Design Tool to emulate.
Transport Type	This setting specifies the transport type being used to connect to the host. You can use Telnet, Telnet Extended, SSH, or NS/VT transport types as well as Secure Socket Layer (SSL) and Transport Layer Security (TLS) security protocols. Telnet is available for 3270,5250, VT, and HP session types, while Telnet Extended is only available for IBM 3270 session types. Choose SSH to connect to a VT host using SSH security protocols. NS/VT is available for HP session types.
Use SSL/TLS	See Using SSL/TLS
Host Name or IP Address	Use this box to identify the host to which you want to connect. You can either select a host from the drop down list or type one into the host name field. When you start the Design Tool without a value for this setting, the Design Tool looks for the Hosts file in the same folder as Wsock32.dll. If you are using third-party TCP/IP software, your Hosts file may be in another location. When it finds a Hosts file, the Design Tool changes the value of this setting.

Options	Description
Port Number	Specifies the host port or socket number that the Telnet session should use. You can enter any number between 0 and 65,535 in this field. You can configure Port in either the Session Setup or View Settings dialog boxes. The default port number for Telnet is **23**.
Device Name	Specifies the Device name (also known as LU name) to which to connect. Supports up to 32 characters. You can map the Device Name setting to a model variable in the Model menu > Variables. This allows a different setting for each server runtime session. If a model variable is mapped to the Device Name setting, the Device Name configured in Session Setup will be used as a default unless changed by a session pool model variable list or a Host Integrator API SetModelVariable method.

More information

Using SSL/TLS

Using SSH

Advanced 5250 Telnet dialog box

You can configure the following options in this dialog box:

Option	Description
Telnet location	Specifies to the host where the connection originated. Can also be used to provide informational messages to the host from the computer. You are not required to enter anything in this box. Supports 260 characters.You cannot change this value while you're connected to a gateway.
Username	Enter your username for the AS/400 Sign On screen if you want to bypass being prompted for it at connection time. You'll also need to enter your password.
Password	Enter your password for the AS/400 if you want to bypass being prompted for it at connection time. You'll also need to enter your username.
Auto SignOn	Select this option to have the transport protocol automatically log you on to the host as soon as you establish a connection in the Design Tool. By default, this check box is not selected.
Send Keep Alive packets	To become aware of connection problems as they occur, you can configure your model to Send Keep Alive packets.

Option	Description
Keep Alive Timeout	The interval between the keep alive requests sent by the Design Tool. The range of values is 1-9999 seconds, and the default is 600 seconds.

Configuring a VT Session

The following options are available in the Session Setup dialog box.

Some of these options are dependent on the transport type you are using, either Telnet or SSH.

Options	Description
Session Type	Specifies the type of session to configure.
Terminal Type	Specifies the terminal (also known as a display station) you want the Design Tool to emulate.
Transport Type	This setting specifies the transport type being used to connect to the host. You can use Telnet, Telnet Extended, SSH, or NS/VT transport types as well as Secure Socket Layer (SSL) and Transport Layer Security (TLS) security protocols. Telnet is available for 3270,5250, VT, and HP session types, while Telnet Extended is only available for IBM 3270 session types. Choose SSH to connect to a VT host using SSH security protocols. NS/VT is available for HP session types.
Host Name or IP Address	Use this box to identify the host to which you want to connect. You can either select a host from the drop down list or type one into the host name field. When you start the Design Tool without a value for this setting, the Design Tool looks for the Hosts file in the same folder as Wsock32.dll. If you are using third-party TCP/IP software, your Hosts file may be in another location. When it finds a Hosts file, the Design Tool changes the value of this setting.
Port Number	Specifies the host port or socket number that the SSH session should use. You can enter any number between 0 and 65,535 in this field. You can configure Port in either the Session Setup or View Settings dialog boxes. The default port number for SSH is **22**.

Options	Description
Device Name	Specifies the Device name (also known as LU name) to which to connect. Supports up to 32 characters. You can map the Device Name setting to a model variable in the Model menu > Variables. This allows a different setting for each server runtime session. If a model variable is mapped to the Device Name setting, the Device Name configured in Session Setup will be used as a default unless changed by a session pool model variable list or a Host Integrator API SetModelVariable method.

See Customizing the VT Terminal for Advanced Telnet and SSH Configurations.

After you connect, you can review these settings in the Connection Properties dialog box.

Configuring an HP Session

On an HP terminal, with the Transport Type set to **Telnet** or **NS/VT**, the following options are available in the Session Setup and New Model dialog box.

After you connect, you can review these settings in the Connection Properties dialog box.

Options	Description
Session Type	Specifies the type of session to configure.
Model ID	Specifies the terminal (also known as a display station) you want the Design Tool to emulate.
Transport Type	This setting specifies the transport type being used to connect to the host. You can use Telnet, Telnet Extended, SSH, or NS/VT transport types as well as Secure Socket Layer (SSL) and Transport Layer Security (TLS) security protocols. Telnet is available for 3270,5250, VT, and HP session types, while Telnet Extended is only available for IBM 3270 session types. Choose SSH to connect to a VT host using SSH security protocols. NS/VT is available for HP session types.
Host Name or IP Address	Use this box to identify the host to which you want to connect. You can either select a host from the drop down list or type one into the host name field. When you start the Design Tool without a value for this setting, the Design Tool looks for the Hosts file in the same folder as Wsock32.dll. If you are using third-party TCP/IP software, your Hosts file may be in another location. When it finds a Hosts file, the Design Tool changes the value of this setting.

Options	Description
Port Number	Select one of the following depending on which transport type you selected in the Session Setup or New Model dialog box; Telnet or NS/VT.The default port number for Telnet is **23**. The default port number for NS/VT is **1570**.

See Customizing HP for advanced HP Telnet configuration information.

More informationi

- Customizing the Terminal
- Setting Up a Connection

4.4.2 Customizing the Terminal

A quick way to customize the terminal is to create a model using a pre-configured settings file provided with Host Integrator. The settings file has defaults appropriate for the terminal session you are modeling.

- Customizing the 3270 Terminal
- Customizing the 5250 Terminal
- Customizing the VT Terminal
- Customizing the HP Terminal

Customizing the 3270 Terminal

You can control various aspects of the Design Tool's behavior as a 3270 terminal. First, configure a 3270 session and then configure any of the options available from the Settings menu to customize your 3270 terminal.

3270 Terminal Setup

- National character set

  The values identify the various host character sets that Host Integrator supports. The Design Tool uses this setting to choose a conversion table that it uses to convert host characters (EBCDIC) into PC characters (ANSI). This setting should match the national character set used by your host system. If it doesn't match, then some characters, such as accents, may not display correctly. See your host documentation for definitions of the characters in each set. The default value is US English.

You can also define a custom code page. After creating the custom code page file, select Custom in the national character set list. A set of examples is included with Host Integrator; see Creating a Custom Code Page for details.

- Country Extended Graphics Code

  When this check box is selected, additional characters are available in the configured National character set. See your host documentation for details. By default this box is selected.

  You can change the settings in the Terminal Setup dialog box if you have opened a model but you are not connected to it.

  - If you have no model open, this dialog box is unavailable.
  - If you are connected to a model, the settings in this dialog box can be viewed but cannot be changed.

Default 3270 keyboard mapping and functions

Function	Keystroke	Environment	Function
Alt Cursor	Alt+6	host	
Attention	Ctrl+F1	host	Interrupts the host application program. Not all host Telnet programs support the Attention key.
Backspace	Backspace	host	
Backtab	Shift+Tab	host	
Clear	Ctrl+F2 or Scroll Lock	host	When you first log on, your host connection is in implicit state, meaning that there is only a single partition. In implicit state, Clear erases all data in the partition. When there are multiple partitions, Clear erases all data in all partitions, and returns the emulator to implicit state. When there are multiple partitions, use the Clear Partition function to clear only the current partition. Most host applications use only a single partition.

Function	Keystroke	Environment	Function
ClearPartition	Ctrl+F4	host	Erases all data in the current partition.
Connect	Alt+C	Design Tool	
Copy	Ctrl+C	Design Tool	
Cursor Blink	Alt+8	host	Changes the blink rate of the cursor, cycling through two options (Blinking Disabled and Windows System).
Cursor Select	Ctrl+F3	host	Simulates a light pen select in the field containing the cursor. Sets or clears the attribute indicating that a field has been modified. You can use the right mouse button in selectable terminal fields to perform a cursor select; in this way, the mouse can be used to simulate a light pen.
Delete Character	Delete	host	Deletes the character the cursor is on.
Delete Word	Alt+Delete	host	Deletes one word of text from an input field.
Disconnect	Alt+D	Design Tool	
Down	Down Arrow	host	
Duplicate	Ctrl+Page Up	host	Inserts a dup character into the buffer. The cursor moves to the first position of the next unprotected field if the screen is formatted, or to row 1, column 1, if the screen is unformatted.
Enter	Enter	host	
EraseEOF	End	host	Erases all data from the cursor location to the end of the current field.

Function	Keystroke	Environment	Function
Erase Input	Alt+F5	host	Erases all unprotected fields in the current partition.
Field Delimit	Ctrl+Alt+F	host	Toggles IBM standard and extended field delimiters on or off during a host session. Toggling can be useful for troubleshooting.
Field Mark	Ctrl+Page Down	host	Inserts a field mark character.
Home	Home	host	
Insert	Insert	host	
Keyclick	Alt+7	host	Toggles the keyclick on or off. When this function is on, the PC emits a low-pitched sound in numeric fields, a high-pitched sound in alphabetic fields, and no sound in protected fields.
Left	Left Arrow	host	
Left, double speed	Alt+Left Arrow	host	
New Line	Shift+Enter	host	
Next Window	Alt+N	Design Tool	
NumLock	Num Lock	host	
PA1	Page Up	host	
PA2	Page Down	host	
PA3	Ctrl+3	host	

Function	Keystroke	Environment	Function
Pan Left	Ctrl+Left Arrow	host	Moves the portion of the current partition visible on the screen so that a different part of the partition is visible. The distance panned is determined by the host upon establishment of the partition. If Host Integrator can display the entire partition on the terminal screen, this function has no effect.
Pan Right	Ctrl+Right Arrow	host	Moves the visible portion of the current partition so that a different part of the partition is visible. The distance panned is determined by the host upon establishment of the partition. If Host Integrator can display the entire partition on the terminal screen, this function has no effect.
Partition Jump	Ctrl+Tab	host	Moves the cursor to the next partition. Continuing to use this function cycles you through all available partitions.
Paste	Ctrl+V	Design Tool	
PF1-PF12	F1-F12	host	
PF13-PF24	Shift+F1+Shift +F12	host	
Reset	Esc	host	
Right	Right Arrow	host	
Right, double speed	Alt+Right Arrow	host	

Function	Keystroke	Environment	Function
Scroll Down	Ctrl+Down Arrow	host	Moves the visible portion of the current partition so that a different part of the partition is visible. The distance scrolled is determined by the host upon establishment of the partition.
Scroll Up	Ctrl+Up Arrow	host	Moves the visible portion of the current partition so that a different part of the partition is visible. The distance scrolled is determined by the host upon establishment of the partition.
Show Command Line	Alt+L	Design Tool	
SysReq	Alt+Print Screen	host	The definition of this key and its values vary by host application.
Tab	tab	host	The definition of this key and values vary by host application.
Toggle Terminal Keyboard	Alt+K	Design Tool	
Toggle Toolbar	Alt+B	Design Tool	
Up	Up	host	

Customizing the 5250 Terminal

You can control various aspects of the Design Tool's behavior as a 5250 terminal. First, configure a 5250 session and then configure any of the options available from the Settings menu to customize your 5250 terminal.

5250 Terminal Setup

- National character set

The values identify the various host character sets that Host Integrator supports. The Design Tool uses this setting to choose a conversion table that it uses to convert host characters (EBCDIC) into PC characters (ANSI). This setting should match the national character set used by your host system. If it doesn't match, then some characters, such as accents, may not display correctly. See your host documentation for definitions of the characters in each set. The default value is US English.

You can also define a custom code page. After creating the custom code page file, select Custom in the national character set list. A set of examples is included with Host Integrator; see Creating a Custom Code Page for details.

You can change the settings in the Terminal Setup dialog box if you have opened a model but you are not connected to it.

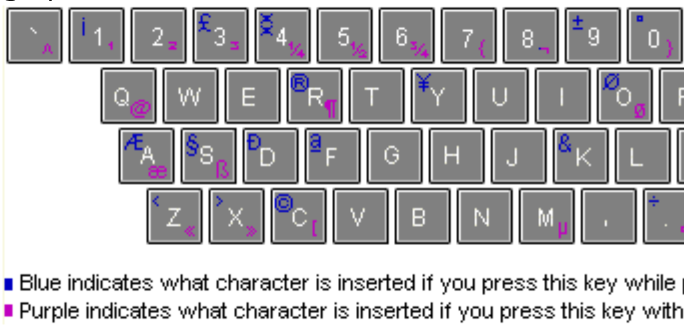
If you have no model open, this dialog box is unavailable.

If you are connected to a model, the settings in this dialog box can be viewed but cannot be changed.

Default 5250 keyboard mapping

Function	Keystroke	Environment	Description
Alt Cursor	Alt+6	host	
Attention	Esc	host	Interrupts the host application program. Workstatio use the Attn key to alert the AS/400 system that a re function (such as Enter) is not being honored. The A allowed whether the keyboard is locked or unlocked
Backspace	Backspace	host	Moves the cursor back to the previous position in wl can be entered. If the cursor is already in the first po input field, Backspace moves back to the last positio preceding input field.
Backtab	Shift+Tab	host	Moves the cursor back to the first position in the cur field. When the cursor isn't on an input field, Backtal to the first position in the previous input field. If the doesn't contain input fields, Backtab moves the curs column 1.
Begin Bold	Ctrl+B	host (word processing)	
Begin Underline	Ctrl+U	host (word processing)	

Function	Keystroke	Environment	Description
Bottom of Page	Ctrl+Down Arrow	host (word processing)	
Carrier Return	Left Ctrl, Right Ctrl	host (word processing)	
Center	Ctrl+C host (word processing)		
Clear	Pause	host	Signals the host to erase all user-entered text from t screen.
Connect	Alt+C	Design Tool	
Delete	Delete	host	Deletes the character at the cursor position. All char the right of the cursor (in the same input field) shift position to the left. Null characters are inserted at th of the field as characters are deleted. If you invoke t when the cursor is not in an input field, the input inh appears in the status line. (In the 5250 status line, th inhibit indicator is the letters II in inverse video.) Pre (Left Ctrl) to clear the symbol and enable input.
Disconnect	Alt+D	Design Tool	
Down	Down Arrow	host	
Down Double	Alt+Down Arrow	host	Moves the cursor down one or more lines. The numb the cursor moves is determined by the value of Vert the Terminal Setup dialog box. The default is two lin
Duplicate	Shift+Insert	host	Uses the data from the equivalent field in the previo the contents of this field. If a field is programmed to duplication, you'll see a symbol at the cursor positio invoke this function. Invoking this function when the not in an input field or not in a field that supports th causes the input inhibit symbol to appear in the stat the 5250 status line, the input inhibit indicator is the inverse video.) Press Reset (Left Ctrl) to clear the syn enable input.

Function	Keystroke	Environment	Description
End Bold	Ctrl+J	host (word processing)	
End of Line	Ctrl+Right Arrow	host (word processing)	
End Underline	Ctrl+J	host (word processing)	
Enter	Enter	host	Transmits any data you have typed in the Terminal w the host application.
Erase EOF	End	host	Erases all data from the cursor position to the end o current field, without moving the cursor.
Erase Input	Alt+End	host	Erases the contents of all input fields on the screen a the cursor to the beginning of the first input field. If has no input fields, Erase Input moves the cursor to column 1 on the screen, and no data is erased.
Extended Graphics	Alt+Right Shift	host	The following special characters are available in exte graphics mode:  ▪ Blue indicates what character is inserted if you press this key while ▪ Purple indicates what character is inserted if you press this key with Host Integrator remains in extended graphics mode press Alt+Right Shift again or Reset (Left Ctrl).
F1-F12	F1-F12	host	
F12-F24	Shift+F1-Shift+F12	host	

Function	Keystroke	Environment	Description
Field Exit	Right Ctrl	host	Moves the cursor out of an input field, inserting null from the current cursor location to the end of the fie cursor moves to the first position of the next input fi use Field Exit in a right-adjust field, the data to the le cursor shifts to the right. The vacated positions are f zeros or blanks as specified by the program, and the advances to the next input field.
Field Minus	Numeric keypad –	host	Moves the cursor out of a signed-numeric or numer inserting a minus sign in the last position of a signed field, or changing the last position in a numeric-only alphabetic character that tells the system that this fi negative value.
Field Plus	Numeric keypad +	host	Except in a signed-numeric or numeric-only field, thi is identical to Field Exit. In a signed-numeric field, th moves the cursor to the next field, removing a minu there is one in the last position. In a numeric-only fie function moves the cursor to the next field, changin position to an alphabetic character that tells the syst this field has a positive value.
Half Index Down	Ctrl+H	host (word processing)	
Half Index Up	Ctrl+Y	host (word processing)	
Help	Scroll Lock	host	Provides help from the system or, when an error cor exists, an explanation of the error condition. See you documentation for information on error codes.

Function	Keystroke	Environment	Description
Hex Mode	Alt+F7	host	Use this function before entering a hexadecimal valu character. To enter a hexadecimal value, type two ch These characters can be 0-9 or a-f. No other input is hex mode. When you press Alt+F7 to put Host Integr mode, a small h appears on the 3488 status line (def to the right of center. Reset (Left Ctrl) causes Host In exit hex mode. When you enter the first character of value, the h in the status line becomes a capital H. If entry mode is selected, pressing Left Ctrl at this poir character out of the buffer but doesn't exit hex mod way of retracting the character you typed. To show t capital H becomes a small h once again. Host Integr expects you to enter two characters for your hex val Left Ctrl again to back completely out of hex mode.
Home	Home	host	Moves the cursor to the first input position on the sc screen contains no input fields, Home moves the cur 1, column 1. If you use this function when the cursor in the first input position on the screen, Host Integra transmits a record backspace aid key instead.
Insert	Insert	host	The insert symbol appears in the 3488 status line (de the 5250 status line, the letters IM are displayed in ir video when Host Integrator is in insert mode.) Chara type when the terminal is in insert mode are inserte cursor position. As you type, existing characters at a right of the cursor position shift one position to the each character you type. There must be a null charac right end of the insert field for each character you ty mode. If you attempt to insert more characters than nulls, an X appears in the status line and input is inh Press Reset Left Ctrl to remove the symbol and enab
Insert Symbols	Ctrl+A	host (word processing)	
Left	Left Arrow	host	
Left Double	Alt+Left Arrow	host	Moves the cursor one or more columns to the left.

Function	Keystroke	Environment	Description
New Line	Shift+Enter	host	Moves the cursor to the first input position on the n there are no input positions on the next line, the cur to the first input position on the next line with an inp you execute Newline when there are no lines betwe current line and the end of the screen that contains it wraps back to the first line in the screen. Newline Tab in that it can be used to drop to the second line line field. Tab always moves to the next field.
Next Stop	Ctrl+N	host (word processing)	
Next Text Column	Ctrl+D	host (word processing)	
Next Window	Alt+N	Design Tool	
PA1-PA3	Alt+F1-Alt+F3	host	
Page Down	Page Down	host	Scrolls down one page in the current host screen (e Roll Up).
Page End	Ctrl+P	host (word processing)	
Page Up	Page Up	host	Scrolls up one page in the current host screen (equi Roll Down).
Plus CR Mode	Alt+F12	host	Execute this function once and Host Integrator show hexadecimal codes in front of each field in the Term These codes indicate the field and display attributes field. If you've configured Host Integrator to use the line (default) or the debug status line, then d appear status line. Execute this function again and Host Inte shows different two-character hexadecimal codes in extended character buffer values for each field. A c a the 3488 or debug status line. Execute this function and Host Integrator shows two-character hexadecim indicating the character attributes for each characte Terminal window. An a appears in the 3488 or debug line.Press Reset Left Ctrl twice to exit Plus CR mode.

Function	Keystroke	Environment	Description
Print	no mapping	host	Sends an image of the screen to the host. How and w image is printed depends on the configuration of th session.
Print Screen	Print Screen	Design Tool	Prints the contents of the Terminal window.
Required Carriage Return	Right Ctrl	host (word processing)	
Required Space	Ctrl+Spacebar	host (word processing)	
Required Tab	Ctrl+Tab	host (word processing)	
Reset	Left Ctrl	host	Press Reset once to exit from various modes and cle errors. Press Reset twice to exit Plus CR mode.
Right	Right Arrow	host	
Right Double	Alt+Right Arrow	host	Moves the cursor one or more characters to the righ
Roll Down	Shift+Down Arrow	host	Scrolls up one page in the current host screen (equiv Page Up).
Roll Up	Shift+Up	host	Scrolls down one page in the current host screen (ec Page Down).
Rule	Alt+Page Down	host	Toggles the rule line into or out of view. The rule line horizontal line, a vertical line, or both, indicating the column containing the cursor.
Show Command Line	Alt+L	Design Tool	
SLP Auto Enter	not mapped	host	The full name of this key is Selector Light Pen Auto E simulates a light pen select at the cursor location.
Stop	Ctrl+S	host (word processing)	

Function	Keystroke	Environment	Description
SysReq	Alt+Print Screen	host	The SysReq key is allowed whether the keyboard is l unlocked. Use this function to: select and start an alt notify the host system that the terminal is ready to s program, or request that the keyboard be unlocked.
Tab	Tab	host	Moves the cursor from the current position to the fi position in the next input field. If there are no input after the current cursor position, Tab moves the curs first character position in the first input field on the s the screen doesn't contain input fields, Tab moves th row 1, column 1 on the screen.
Tab Advance	Ctrl+T	host (word processing)	Moves the cursor from the current position to the n Tab stop. If there are no Tab stops in the current fiel Advance moves the cursor to the first Tab stop in the field on the screen. Tab Advance is defined for Text only.
Toggle Toolbar	Alt+B	Design Tool	
Toggle Terminal Keyboard	Alt+K	Design Tool	
Top of Page	Ctrl+Up	host (word processing)	
Test Request	Alt+Scroll Lock	host	Signals the host to enter test request mode, a set of programs created by IBM and provided by the AS/40 function is only available in the sign-on screen.
Up	Up	host	
Up Double	Alt+Up	host	Moves the cursor up one or more lines.

Function	Keystroke	Environment	Description
Word Underline	Ctrl+W	host (word processing)	

Customizing the VT Terminal

You can control various aspects of the Design Tool's behavior as a VT terminal. First, configure a VT session and then configure any of the options available from the Settings menu to customize your VT terminal.

Configuring VT Emulation Options

From the Settings menu, click Terminal. The Emulation tab includes the following:

- Host Character Set

  The Host Integrator VT terminal supports the following character sets:

  DEC Supplemental

  ISO ISO Latin-1 (ISO-8859-1) This is the ISO-8859-1 character set

  ISO Latin-9 (ISO-8859-15)

  PC English (437)

  PC Multilingual (850)

  Windows Latin (1252)

  The default value for the Host character set depends on the type of terminal you are emulating. This setting reflects the current terminal state of VT Host Character Set, which can be changed by the host. The associated default setting, saved with the model is VT Host Character Set Default.

  When a soft reset is performed, or when initially connecting, the default host character set is used.

  The current terminal state for the host character set can be altered by invoking the DECSTR sequence. The Host character set may also be specified by the Select Character Set (SCS) sequence.

- Terminal ID

The setting in this list determines the response that Host Integrator sends to the host after a primary device attributes (DA) request. This response lets the host know what terminal functions it can perform. Host Integrator's response for each Terminal ID is exactly the same as the VT terminal's response; some applications may require a specific DA response. This terminal ID setting is independent of the Terminal type setting.

The options are VT100, VT220, VT320, VT420, and VT52.

This setting reflects the current terminal state of VT Terminal ID, which can be changed by the host. The associated default setting, saved with the model, is VT Terminal ID Default.

- Online

  Host Integrator is always in either remote or local mode. Keeping this check box selected means that Host Integrator can communicate with the host as a terminal; this is known as remote mode. Host Integrator transmits each character typed from the keyboard to the host. As Host Integrator receives characters from the host, it displays them on the screen.

  When Host Integrator is in remote mode, but is not connected to a host computer, characters typed at the keyboard do not appear on the screen.

  When you clear this check box, Host Integrator can no longer communicate with the host; this is known as local mode. In local mode, Host Integrator does not attempt to communicate with a host computer. Characters entered from the keyboard appear on the screen, but are not transmitted to the host; nor is any data from the host (for example, notification of a mail message) received by Host Integrator.

- Newline

  Selecting this check box causes Host Integrator to send both a carriage return and linefeed when you press Enter (known as new line mode). When Host Integrator receives a linefeed, form feed, or vertical tab, it moves the cursor to the first column of the next line. When this check box is left cleared (linefeed mode), the Return key sends only a carriage return. A linefeed, form feed, or vertical tab received from the host moves the cursor down one line in the current column.

  If lines on the display is being overwritten (that is, the host is not sending a linefeed along with a carriage return), select this check box. If the New line check box is selected but the host does not expect to receive a linefeed with each carriage return, lines will be double-spaced on the display.

  This setting reflects the current terminal state of VT New Line, which can be changed by the host. The associated default setting, saved with the model, is VT New Line Default.

- Autowrap

  Selecting this check box causes characters to wrap to the next line automatically when the cursor reaches the right margin of the display. This setting is different from the VAX host's terminal wrap characteristic, which is set with this DCL command: SET TERMINAL/[NO]WRAP

  The host command determines whether characters wrap automatically when they reach the maximum terminal width set by the host's SET TERMINAL/WIDTH command. Host Integrator's

Autowrap option, in contrast, determines whether characters wrap when they reach the right margin of the display. When Host Integrator is communicating with the host:

If...	**Then...**
If terminal wrap is set on the host	Characters wrap when they reach the maximum terminal width, regardless of the setting of the Autowrap check box.
If terminal wrap is not set and the Autowrap check box is selected	Characters wrap at the right margin of the display
If terminal wrap is not set and the Autowrap check box is cleared	Characters never wrap when they reach the right margin of the display. New characters overwrite the character at the right margin until a carriage return is entered

When Host Integrator is not communicating with the host (that is, the Online check box is not selected), the Autowrap check box works as described in the last two items above, as if the host's terminal wrap characteristic is not set.

- National Replacement Character Set

  This setting specifies a set of character translations that occur between the local computer and the host in 7-bit mode. This setting affects how characters entered from the keyboard or from a local file are transmitted to the host, and how characters sent from the host are written to local files, to the screen, or both.

  Translations are not performed unless the Use NRC setting is set to Yes.

  This setting reflects the current terminal state of the National Replacement Set, which can be changed by the host. The associated default setting, saved with the model, is VT National Replacement Set Default.

- Use NRC (7-bit) Set

  This setting specifies whether the translations specified by the National Replacement Set setting should be performed.

  This setting reflects the current terminal state of VT Use NRC, which can be changed by the host. The associated default setting, saved with the model, is VT Use NRC Default.

- Answerback message

  This setting allows you to enter an answerback message if the host expects an answer in response to an ENQ character.

  This setting reflects the current terminal state of VT Answerback Message, which can be changed by the host. The associated default setting, saved with the model, is VT Answerback Message Default.

Select the *Insert special characters* check box to include escape sequences and ASCII control codes in the message.

Select *Auto answerback* to specify whether the answerback message is automatically sent to the host after a communications line connection.

- User Features Locked

  This setting specifies whether certain features can be changed by the host. When this setting is set to Yes, the following properties cannot be changed by the host: - Tab stops - The Keyboard Locked setting which specifies whether or not the keyboard is locked (that is, it cannot be used). - The Inverse Video setting which specifies whether the foreground and background colors for screen attributes are reversed.

- User-defined keys locked

  When this setting is set to Yes, the Host Integrator locks the user keys to prevent the host from clearing or redefining them. When user-defined keys are locked, the only way they can be redefined is to first unlock them, and the only way to unlock them is by selecting No. This setting is not saved to a Host Integrator settings file.

  If you want to define new user keys or allow the host to define them, the keys must be unlocked. If user keys are locked and an application tries to redefine a key using a DECUDK sequence, Host Integrator ignores the sequence.

  This setting reflects the current terminal state of VT User-defined Keys Locked, which can be changed by the host. The associated default setting, saved with the model, is VT User-defined Keys Locked Default.

Advanced VT SSH Options

- User authentication

  Click in the box next to any authentication method to clear or enable that method. You must select at least one authentication method. You can use the arrows to specify your order of preference. The first method you select that is supported by the server is used.

- Cipher list

  Use this list to specify the ciphers you want to allow for connections to the current host. When more than one cipher is selected, the SSH client attempts to use ciphers in the order you specify, starting from the top. To change the order, select a cipher from the list, then click the up or down arrow. The cipher used for a given session is the first item in this list that is also supported by the server.

- HMAC list

  Specifies the HMAC (hashed message authentication code) methods you want to allow. This hash is used to verify the integrity of all data packets exchanged with the server. When more than one

HMAC is selected, the SSH client attempts to negotiate an MAC with the server in the order you specify, starting from the top. To change the order, select an HMAC from the list, then click the up or down arrow.

- Key exchange algorithms

  Specifies which key exchange algorithms the client supports, and the order of preference. In some cases, you may need to change the order of the key exchange algorithms to put DH Group14 SHA1 ahead of the other values. This is required if you want use the hmac-sha512 HMAC, or if you see the following error during key exchange: "Unable to exchange encryption keys."

- Host key algorithms

  Specifies the host key type the client will accept from the server. You must select at least one host key type. The SSH client will use a key type from the server in the order you specify, starting from the top. To change the order, select an algorithm from the list, then click the up or down arrow.

- Keep alive

  When **Keep Alive** is selected, Host Integrator sends NOOP messages to the server through the secure tunnel at the specified interval. Use this setting to maintain the connection to the server. Use **Interval in seconds** to specify how frequently server alive messages are sent. If this setting is not enabled, the SSH connection will not terminate if the server dies or the network connection is lost.

  The SSH **Keep Alive** setting is not related to the TCP keep alive setting that can be set in the Windows registry to keep all TCP/IP connections from being timed out by a firewall. To change the TCP/IP keep alive behavior, you need to edit the Windows registry.

- Enable compression

  When Enable compression is selected, the client requests compression of all data. Compression is desirable on modem lines and other slow connections, but will only slow down response rate on fast networks.

- Only show FIPS validated values

  Select this to filter the Cipher and HMAC lists to show only FIPS validated values. SSH only uses the values you configure. If you choose a non-FIPS value while running in FIPS mode an error message asking to you to specify a valid cipher displays.

  Federal Information Processing Standards (FIPS) are guidelines established by the United States government to standardize computer systems. To use FIPS 140-2 validated encryption, in a Windows environment, you must first define an environment variable, VHI_FIPS = 1.

Advanced VT or HP Telnet

You can configure the following options in this dialog box:

- Terminal ID

When Host Integrator connects to a Telnet host, the Telnet protocol negotiates a Telnet terminal ID with the host. In general, this negotiation results in the selection of the correct terminal ID. However, if you are having trouble running a host application, the negotiation between Telnet and the host could be the issue.

This setting determines:

which screen control sequences the host sends to Host Integrator to format the screen.

the position of the cursor

what characters to display in a host application

To override Host Integrator's election of a terminal, select a terminal ID from the list (because this list box is editable, you can just type in any value). If you enter a terminal ID that the host does not recognize, Host Integrator reverts to a list of default values until one is found that the host supports. The default terminal ID values are VT100, VT 220, VT 320, VT 420 and VT52 and HP and HP2392A. The terminal type and the settings are independent of each other. If you enter a terminal ID string, it may be up to 40 characters long taken from a set of uppercase letters, digits, and the two punctuation characters, hyphen and slash. It must start with a letter and end with a letter or digit.

- Location (Optional)

  Specifies to the host where the connection originated. Can also be used to provide informational messages to the host from the PC. Usage conventions vary by site. Supports up to 260 characters. You cannot change this value while you're connected to a gateway.

- Send LF after CR

  A "true" Telnet host expects to see a CrNu (carriage return/null) character sequence to indicate the end of a line sent from the Design Tool. There are some hosts on the Internet that are not true Telnet hosts, and they expect to see a Lf (linefeed) character following the Cr at the end of a line. If you're connecting to this type of Telnet host, select this check box.

- Initiate Option Negotiation

  Specifies whether certain connection options, including whether to always request a binary mode connection, should be negotiated when the Telnet connection is established. Connections to some hosts on the Internet are expedited if this check box is cleared so that the Design Tool does not attempt to initiate negotiations for Telnet options.

- Request Binary

  Telnet defines a 7-bit data path between the host and the terminal (Host Integrator). This type of data path is not compatible with certain national character sets. Fortunately, many hosts allow for 8-bit data without zeroing the 8th bit, which resolves this problem. Select this check box to force the host to use an 8-bit data path.

- Set Host Window Size

Sends the number of rows and columns to the Telnet host whenever they change. This enables the Telnet host to properly control the cursor if the window size is changed.

- Ctrl-break Character

  Specifies what happens when you press the following keys:

  ATTN key (Ctrl-F1 by default) in a 3270 session

  Ctrl-break in a VT or HP session

  Options to send to the host are:

  Telnet Abort Output

  Telnet Break

  Telnet Interrupt Process

- Send Keep Alive Packets

  In some cases, Host Integrator may become aware of Telnet communication problems only after a significant delay or when it attempts to send data to the host. This can cause problems if you enter a large amount of data on one screen or if you keep your connection open during periods of inactivity.

  To become aware of connection problems as they occur, you can configure your model to send Keep Alive packets. Four methods are available:

  None - No keep alive packets are sent (default)

  Send NOP Packets - Periodically a No Operation (NOP) command is sent to the host. The gateway and host are not required to respond to these commands, but the TCP/IP stack can detect if there was a problem delivering the packet.

  System - The TCP/IP stack keeps track of the host connection. This method requires less system resources than Send Timing Mark Packets or Send NOP Packets, but most TCP/IP stacks send Keep Alive packets infrequently.

  Send Timing Mark Packets - Periodically a Timing Mark Command is sent to the host to determine if the connection is still active. The gateway or host should respond to these commands. If a response is not received or there is an error sending the packet, it will shut down the connection. To view the average amount of time has waited for a response to a Timing Mark Command in the Design Tool, open the View Settings dialog box and select the Telnet Average Keep Alive Roundtrip setting.

- Keep Alive Timeout

  The Keep Alive Timeout is the interval between the keep alive requests sent by the Design Tool. The range of values is 1-9999 seconds, and the default is 600 seconds.

- Local Echo

Controls how the Design Tool responds to remote echo from a Telnet host: - Automatic - (Default) Attempts to negotiate remote echo, but does what the host commands. - Yes - Negotiates local echo with the host, but always echoes. - No - Negotiates remote echo with the host, but does not echo.

VT Control Functions

Control functions cause Verastream to perform certain actions, such as move the text cursor, add a line of text, assign character attributes, and change character sets. Typically, the host application sends control functions to Verastream to perform the desired actions. There are three symbols used in this section that describe a sequence:

Symbol	Description
ESC	Stands for the Escape character, and begins an escape sequence
CSI	Stands for the Control Sequence Introduction character. When Verastream receives this character, it recognizes the string that follows as a control sequence
DCS	Stands for the Device Control String character, and begins a device control sequence

Control Function Notation

The following notation is used throughout this section:

Most control functions have a mnemonic identifier. You will never need to enter the mnemonic; it’s simply a convenient word to help you remember the name of the control function. For example, DECCOLM is the Digital mnemonic for setting the number of display columns.

Control functions are case sensitive, and must be typed exactly as shown.

Where appropriate, a slash is used through a zero (Ø) to distinguish it from the uppercase letter O.

Note the difference between a lowercase L (l) and the number 1 (1). References are occasionally given to a character’s decimal value in the character set charts. For example, the space character is ASCII decimal 32. Control characters are always in the same positions in the charts, and can be located uniquely by their decimal value; the word “ASCII” is omitted.

This topic shows control functions in their 8-bit format. You can always use the 7-bit equivalent escape sequence to represent the 8-bit control, as shown in the Recognized C0 Control Characters Table.

Parameters you supply for a sequence are enclosed in angle brackets. For example, when using the `CSI<n>C` control function, replace `<n>` with the number of spaces you want the cursor to move. If `<n>` is omitted, the default is used. For most sequences, the default is 1 (when there is not a corresponding Ø sequence).

Numeric parameters are represented by ASCII strings. For example, in the sequence CSI10;13H the numbers are the strings 10 and 13, not the decimal values 10 (LF character) and 13 (CR character). Numeric parameters are constrained to the range 0–9999, inclusive. Any numeric parameter with a value greater than 9999 is interpreted as 9999. The plus sign (ASCII decimal 43) and the minus sign (ASCII decimal 45) are not within the range of legal control characters. Therefore, signed numbers, for example "+10" or "-12," cause a control sequence or a device control string to be rejected. Do not include signs with numeric parameters.

Single-Character Control Functions

A single-character control function is made up of one control character (in contrast to multiple-character control functions). There are two sets of single-character control functions available on VT200, VT300, and VT400 terminals:

- C0 control characters have decimal values 0–31. Verastream supports only the C0 characters shown in the **Recognized C0 (7-Bit) Control Characters Table**. Each C0 character is encoded in 7 data bits, with the high-order bit always 0. C0 codes, therefore, can be used in both 7-bit and 8-bit operating environments. LF and CR, for example, are single-character C0 control functions.
- C1 control characters have decimal values 128–159. Verastream supports only the C1 characters shown in the Recognized C1 (8-Bit) Control Characters Table. Each C1 character is encoded in 8 data bits, with the high-order bit always 1. C1 characters provide a few more functions than C0 characters, but can only be used directly in an 8-bit operating environment. DCS, for example, is a single-character C1 control function.

In 7-bit operating environments, C1 characters must be converted to a two-character escape sequence equivalent.

Recognized C0 (7-Bit) Control Characters Table

The following table lists the C0 control characters that Verastream recognizes. C0 controls can be used in both 7-bit and 8-bit environments. To represent C0 controls in a Verastream macro, use the Chr$(`<n>`) syntax, where `<n>` is the decimal value of the C0 control. See the "Decimal" column in the following table:

Name	Character	Octal	Keystroke	Action
Null	NUL	0	^@	Ignored when received
Enquiry	ENQ	5	^E	Transmits the answerback message
Bell	BEL	7	^G	Sounds a bell

Name	Character	Octal	Keystroke	Action
Backspace	BS	10	^H	Moves cursor left one position on the current line
Horizontal tab	HT	11	^I	Moves cursor to the next tab stop, or to the right margin.
Linefeed	LF	12	^J	Causes a linefeed or new line operation.
Vertical tab	VT	13	^K	Same as linefeed
Form feed	FF	14	^L	Same as linefeed
Carriage return	CR	15	^M	Moves the cursor to the left margin of the current line
Shift out (locking shift 1)	SO	16	^N	Maps the G1 character set into GL. G1 is designated by using a Select Character Set (SCS) sequence.
Shift in (locking shift 0)	SI	17	^O	Maps the G0 character set into GL. G0 is designated by using a Select Character Set (SCS) sequence.
Device control 1 (XON)	DC1	21	^Q	Continues sending characters when transmit is set to Xon/Xoff.
Device control 3 (XOFF)	DC3	23	^S	Stops sending characters when transmit is set to Xon/Xoff.
Cancel	CAN	30	^X	Cancels the sequence when received during an escape or control sequence.

Name	Character	Octal	Keystroke	Action
Substitute	SUB	32	^Z	Same as cancel;displays a backwards question mark.
Escape	ESC	33	^[	Introduces an escape sequence and cancels any escape sequence or control sequence in progress.
Delete	DEL	177	(none)	Ignored when received.

Recognized C1 (8-Bit) Control Characters

In 8-bit environments, C1 controls can be sent directly. In a 7-bit environment, you can send an 8-bit C1 control character by converting it to an equivalent 7-bit escape sequence. The 8-bit controls are single character codes (such as CSI), whereas their 7-bit equivalents are two-character sequences (such as ESC[).

To form an equivalent 7-bit escape sequence from an 8-bit control character:

Subtract the hexadecimal value 40 from the C1 control code’s value.

Precede the result with the ESC character.

For example, the IND character (decimal 132) has a hexadecimal value of 84. To convert IND to a 7-bit equivalent, first subtract hexadecimal 40: 84 hex - 40 hex = 44 hex. Hexadecimal 44 is the letter D. Therefore, to represent the IND character in a 7-bit environment, you would use ESCD.

These are the C1 control characters that Verastream recognizes:

Name	Character	Hex	7-Bit Equiv.	Action
Index	IND	84	ESC D	Moves cursor down one line in same column, and scrolls at bottom margin.
Next line	NEL	85	ESC E	Moves cursor to first position of next line, and scrolls at bottom margin.
Horizontal tab set	HS	88	ESC H	Sets a tab stop at the cursor position.
Reverse index	RI	8D	ESC M	Moves cursor up one line in the same column, and scrolls at top margin.
Single shift 2	SS2	8E	ESC N	Maps G2 into GL for the next character only.
Single shift 3	SS3	8F	ESC O	Maps G3 into GL for the next character only.
Device control string	DCS	90	ESC P	Introduces a device control string.
Control sequence	CR	9B	ESC [	Introduces a control sequence.
String	ST	9C	ESC \	Ends a device control string.

Multiple-Character Control Functions

The C0 and C1 codes listed in the Recognized C0 (7-Bit) Control Charactersand Recognized C1 (8-Bit) Control Characters tables are single-character control functions, performing simple functions.

Multiple-character control functions, in contrast, can perform many more functions than the C0 and C1 controls, and are formed by a sequence of characters. There are three types of multiple-character control functions, introduced by the following single-character C0 and C1 controls:

Escape sequences - introduced by the ESC character

Control sequences - introduced by the CSI character

Device control strings - introduced by the DCS character

Multiple-character control functions include both control characters and normal ASCII text, such as letters, numbers, and punctuation. For example, the multiple character device control string DCSØ! u%5ST assigns a user-preferred supplemental character set.

- Escape Sequences

  An escape sequence begins with the C0 character ESC (decimal 27). To enter the ESC character in the Design Tool, press the ESC key. After receiving an ESC character, Verastream interprets the next characters as part of the sequence.

- Control Sequences

  A control sequence begins with the C1 control character CSI (decimal 155). To enter the CSI character in the Design Tool, press the ESC key plus the [key.

  Control sequences usually include variable parameters. The format is:

  `CSI P...P I...I F`

  `P...P` - Zero or more parameters. You can have up to 16 parameters per sequence, using a semicolon (;) to separate each one.

  `I...I` - Zero or more intermediate characters.

  `F` - The final character indicating the end of the control sequence.

  For example, the following control sequence sets the scrolling region, where the top margin is at line 7 and the bottom margin is at line 18:

  `CSI7;18r`

  In this example, 7 and 18 are the parameters and r is the final character. This sequence does not have any intermediate characters.

- Device Control Strings

  A device control string begins with the C1 control character DCS (decimal 144). To enter the CSI character in the Design Tool, press the ESC key plus the P key. Device control strings always include a data string. The format is:

  `DCS P...P I...I F <data string> ST`

  `P...P` - Zero or more parameters. You can have up to 16 parameters per sequence, using a semicolon (;) to separate each one.

  `I...I` - Zero or more intermediate characters.

  `F` - The final character indicating the end of the device control string.

  `<data string>` - A data string of zero or more characters. Use a semicolon to separate individual strings. The particular range of characters included in a data string is determined by the individual device control string. Any character except ST (decimal 156) can be included in a data string.

  `ST` - The string terminator. Type `Chr$(27)& "Ë` for 7-bit mode, or `Chr$(156)` for the 8-bit equivalent. (You can also press `e-\` in the terminal window.)

- APC, OSC, and PM

  The application program command (APC), operating system command (OSC), and privacy message (PM) are also C1 controls. However, VT terminals—and Verastream—ignore them. The controls have the same format as device control strings and end with an ST. When Verastream receives an APC, OSC, or PM introducer, it discards all following characters until receiving a string terminator (so the whole sequence is discarded).

- Interrupting Control Sequences

You can use the following C0 control characters to interrupt a control function or recover from an error:

`ESC` Cancels a sequence in progress, and begins a new sequence.

`CAN` Cancels a sequence in progress. Verastream interprets the characters that follow the CAN character as usual.

`SUB` Same as CAN, except displays a backwards question mark.

Using Nonprintable Characters

Some host application models require you to use nonprintable characters within strings. For example, they may be required when waiting for cursor positioning sequences sent from a VT or HP host using the Host Communication string event WaitForCommString, or sending control sequences via the TransmitANSI command. To represent nonprintable control characters when defining your model in Host Integrator dialog boxes, use the standard C programming convention of encoding each character as a \nnn sequence, where nnn is the octal numeric value of the ASCII character, or use one of the \x sequences that have been defined for the more common control characters. For example, the escape character (0x1B), which prefixes VT cursor positioning sequences, is encoded as `\033`; carriage returns can be encoded using `\r`.

When a model file is read by Host Integrator, the \nnn sequences are parsed into their literal nonprintable character equivalents. Because backslashes () are interpreted as the beginning of a nonprintable character sequence, Host Integrator represents literal backslashes in strings with two backslashes. For example, if you want to wait for the display string "Enter a backslash () to continue" in an operation, what you will see in the Operation Edit dialog box for WaitForDisplayString is "Enter a backslash (\) to continue". If you modify a model file in XML format or the model file itself, you must follow this syntax to embed literal backslashes within string arguments.

Alphabetic Sequence	Equivalent Octal	Also Known As
`\a`	`\007`	Control-G (BEL) Bell
`\b`	`\010`	Control-H (BS) Backspace
`\t`	`\011`	Control-I (TAB) Horizontal tab
`\n`	`\012`	Control-J (LF) Linefeed
`\v`	`\013`	Control-K (VT) Vertical tab
`\f`	`\014`	Control-L (FF) Formfeed

Alphabetic Sequence	Equivalent Octal	Also Known As
`\r`	`\015`	Control-M (CR) Carriage return

Default VT Keyboard Mapping

To find a VT keystroke's default mapping, click the key and click the Default button and note the corresponding Action in the lower half of the dialog box. Click Cancel if you do not want to revert the key's mapping to its default value.

VT Screen Setup

These settings reflect the current terminal state, which can be changed by the host.

- Display columns

  This setting specifies the number of columns displayed in the Terminal window.

- Display columns

  This box sets the number of lines on the display, not including the status line. The maximum number of rows you can enter depends on the display resolution.

  When you change the number of rows, characters are scaled vertically to fit the desired number of rows in the Terminal window. The display is erased before the new setting takes effect.

- Status Line Type

  The VT status line at the bottom of the Terminal window is a one-line display that shows information about the current session. When the Status line is set to Indicator it shows the current row and column of the text cursor.

  When the Status line is set to Host Writable, host applications can display messages on it. The control function DECSASD selects whether the Terminal window or the status line is the active area for displaying text.

- Interpret controls

  The VT terminal character set includes 65 control characters with decimal values 0-31 and 127-159. This option determines whether Host Integrator should interpret or display control characters.

Customizing the HP Terminal

You can control various aspects of the Design Tool's behavior as an HP terminal. First, configure an HP session and then configure any of the options available from the Settings menu to customize your HP terminal.

See Configuring Sessions for information on how to configure the HP terminal.

A quick way to customize the terminal is to create a model using a pre-configured settings file provided with Host Integrator. The settings file has defaults appropriate for the terminal session you are modeling.

Configuring HP Emulation Options

These settings reflect the current terminal state, which can be changed by the host. The associated default setting, saved with the model, is HP Field Separator Default.

Emulation Options	Description
Field Separator	When the Host Integrator is transmitting in block, page, and format modes, it sends a field separator character after each field of the formatted screen except the last one. The value selected here specifies which ASCII character is sent. The values are ASCII decimal 0-127. The default is US (ASCII decimal 31).
Block Terminator	Under certain conditions, the Host Integrator transmits a block terminator character at the end of each block of data transmitted. The value selected here specifies which ASCII character is sent to indicate that the end of the block has been reached. The values are ASCII decimal 0-127. The default is RS (ASCII decimal 30).
Return Definition	Type a character in these boxes to be generated whenever Return is pressed. If the second character is a space, only the first character is generated. The values are ASCII decimal 0-127. The default is CR (ASCII decimal 13).
Host Prompt	An HP 3000 sends a DC1 character to indicate that it is ready to accept a line or block of characters. This character is sent immediately after the MPE colon prompt is sent. This list allows you to change which character is expected. Most hosts either use the DC1 (^Q) character or no prompt (which shows up simply as a space). Select the appropriate host prompt from this list (turn on DISPLAY FUNCTIONS to see the control codes sent by the host before changing this value). The values are ASCII decimal 0-127. The default is D1 (ASCII decimal 17)

Emulation Options	Description
Start Column	For every line in display memory, the Host Integrator attempts to remember the leftmost column that was entered from the keyboard, as opposed to that received from datacomm. This way, the Host Integrator can distinguish the host prompt portion of each line from the user-entered portion. This information is used when you enable LINE MODIFY or MODIFY ALL to determine the leftmost column that should be transmitted to the host when you press Enter or Return. Under some circumstances, it is impossible for the Host Integrator to tell which column was the first user-keyed column; when that happens, it uses the value you enter in this box to determine the leftmost column to be transmitted. When Display Columns are set to 80, enter a value from 0 to 79. When you're in 132-column mode, enter a value from 0 to 131. The values are 0-131. The default is 0.
Host Character Set	The available host character sets for HP terminals are HP Roman 8 and HP Roman 9. The default is HP Roman 8.
Online (Remote Mode)	Specifies whether Host Integrator transmits each number, letter, or special character typed at the keyboard to the host. As characters are received from the host, Host Integrator displays them on your screen.
Inhibit EOL Wrap	When cleared, the Host Integrator automatically returns the cursor to the left margin in the next line when the cursor reaches either the right margin or the right screen edge. When selected, the cursor is not automatically advanced when you reach the right margin. As you type additional characters, each one overwrites the character at the right margin until you explicitly move the cursor by pressing Return or using an arrow key.
Enq/Ack	Some HP 3000 and HP 1000 computers use a form of handshaking called Enq/Ack (Enquire/Acknowledge) to prevent the terminal (or in this case, the Host Integrator) from falling too far behind the host system and losing data. With Enq/Ack pacing, the host system sends 80 characters followed by an ASCII Enq character and stops transmitting. When the Host Integrator has processed all of the characters preceding the Enq, it sends an ASCII Ack character, which tells the host it is ready for more data. "Classic" HP 3000 architecture uses this form of pacing. MPE\iX systems do not; the Enq/Ack setting is disregarded in this environment.

Emulation Options	**Description**
Use Host Prompt	Clear this check box if you want the Host Integrator to ignore the host prompt. Clearing the Use host prompt check box has the same effect as selecting the Inhibit Handshake and Inhibit DC2 check boxes. Ignoring the host prompt forces the Design Tool to behave as though both inhibits are on, thus preventing handshaking. Over an X.25 network, this prevents communications problems caused by applications that use handshaking. When the Use host prompt check box is cleared, the Design Tool always responds to a primary status request from the host that both Inhibit handshake and Inhibit DC2 are enabled. This can affect a host application that explicitly changes one of these inhibits.
Inhibit Handshake	This check box, along with Inhibit DC2 and some other factors, determines the type of handshaking that precedes each block transfer of data from the Host Integrator to the host system. When selected, the DC1 handshake for block transfers is inhibited.
Transmit Functions	Most keyboard keys have an associated ASCII character. Several keys perform functions for which there is no character defined; for example, Home and PgUp. Certain host software programs, such as HP Slate, need to be informed when you press one of these non-ASCII keys. Selecting this option signals the Host Integrator to inform the host system whenever you press one of these keys. When this check box is selected and the Design Tool is operating in character/remote mode, each time you press one of those keys the associated escape sequence is transmitted to the host. Most applications that require this feature automatically send the escape sequences to enable and disable the feature, so you probably will never need to enable it manually.
SPOW	Ordinarily, the Spacebar overwrites and erases existing characters. When the SPOW (SPace OverWrite) check box is selected, spaces entered from the keyboard (not spaces echoed from the host), move the cursor over existing characters, but do not overwrite them with spaces: the SPOW latch is turned on by a carriage return and off by a linefeed, tab, or home up.

Emulation Options	Description
Block Transfer Unit	When operating in block mode, a block of one or more characters is transmitted when you press Enter or when the host requests a block transfer from terminal memory. This option determines how much data Host Integrator transmits on each block transfer. When set to Line, data is transmitted one line at a time, or one field at a time in format mode. When set to Page, data is transmitted one page at a time.

Configure HP Keyboard Options

To find a HP keystroke's default mapping, click the key and click the Default button and note the corresponding Action in the lower half of the dialog box. Click Cancel if you do not want to revert the key's mapping to its default value.

HP Terminal Function Key Configuration

The Function Keys tab lets you select which set of key labels are displayed along the bottom of your screen, and lets you customize eight user keys. A user key definition consists of the following:

A 16-character label (eight characters per line)

A string of up to 80 characters that is produced when the key is pressed

An attribute determining how the Design Tool processes the string when the key is pressed

To reconfigure a function key:

1. Select the key to configure by clicking it with the mouse or by pressing the key on the keyboard.
2. Enter a key string of up to 80 characters to be processed when you press the function key in the Key string box.

   These characters may be handled as if they were entered into the keyboard, or to specify Host Integrator commands. Use the Home, End, arrow keys, or the mouse, to quickly edit a long string. To delete characters, use Backspace or Delete.

   To include escape sequences and ASCII control codes in the user key string, select **Insert special characters**. If you're using the Tab key to tab through the dialog box fields, you must clear the Insert special characters check box; otherwise you'll insert the ASCII tab character each time you press the Tab key.

   The following table shows some examples of keys and key combinations that create certain escape sequences (shown by the two-letter mnemonic that appears on your screen):

Press this...	To include this sequence
Enter	CR
Tab	HT
Backspace	BS
Esc	EC
Ctrl+Q	D1
Ctrl+S	D3
Ctrl+E	EQ
Ctrl+X	CN

Note

To remove a special character, you must first clear the **Insert special characters box.**

1. In the two lines provided in the **Label** box, type up to eight characters per line for the user key label. Use spaces to position and center text in the label. You can also use the editing keys (for example, Ins and Del).
2. Assign an attribute to the user key in the **Attribute** box

   Normal — The key string is treated exactly as if it had been typed from the keyboard; a carriage return is not automatically transmitted.

   If the Host Integrator is in local mode, the string displays on the screen and any embedded escape sequences are executed locally. In remote mode with local echo off, the string is transmitted to the host. It executes and displays only if the host system echoes. For example, use this attribute to store commands that have changing parameters (like the phone number needed to dial different modems).

   Local — The key string is executed locally, and not transmitted to the host. For example, you could assign a user key to home the cursor and clear the display.

   Transmit — In remote mode, the Host Integrator sends the key string to the host after completing a block transfer handshake, automatically transmitting a carriage return. For example, use this attribute to store commonly used commands (such as program run commands) in a user key. In local mode, pressing a user key with this attribute has no effect.
3. If you don't want the Host Integrator to display the function keys and labels along the bottom of the screen, clear the **Show function keys** check box.

Advanced HP Telnet

See Advanced VT or HP Telnet in the Customizing VT section.

Port Number

Select one of the following depending on which transport type you selected in the Session Setup or New Model dialog box:

- Telnet port

  Specifies the host port or socket number that the Telnet session should use. You can enter any number between 0 and 65,535 in this field. You can configure Port in either the Session Setup or View Settings dialog boxes.The default port number for Telnet is 23.

- NS/VT portion

  Specifies the host port or socket number that the NS/VT session should use. You can enter any number between 0 and 65,535 in this field. You can configure Port in either the Session Setup or View Settings dialog boxes. The default port number for NS/VT is 1570.

  !!! note NS/VT is a proprietary HP protocol that connects to HP3000 hosts only.

Identifying Commands

A command can be a host keystroke, a Design Tool command, or any combination of these.

Creating a Custom Code Page

You can create a custom code page for 3270 and 5250 models by modifying any of the sample code page files provided with Host Integrator. By default, these examples are located in `<VHI folder>\codepage` folder for the Design Tool.

The basic steps for creating and implementing a custom code page are:

Create a text file that includes the customization. The easiest way to create this file is to open an appropriate code page sample and make a modification.

Save the modified file as custom.codepage in the same folder as the code page samples for the Design Tool. For a server version, add custom.code page in a codepage directory beneath the server directory.

After saving custom.codepage, restart the Host Integrator server or Design Tool.

Editing a Code Page

A code page is a text file that includes character mappings or property values that differ from the default set (US English).

The property values at the top include the character set ID, the IBM code page, and, in the case of a 5250 model, the keyboard type. Additional information for specifying the Windows font when displaying international characters in the Design Tool can be included. The CanadianFrench.codepage.sample is used in the example below.

Example

```
//======================================================================
// VHI Custom 3270/5250 character translation table sample file.
// IBM EBCDIC codepage (CanadianFrench)
//
// Note: To enable this national character set for 3270 and 5250 sessions,
// copy and rename this file to "custom.codepage"
//
// Copyright...... Copyright (c) 2024 Rocket Software, Inc. or its
affiliates.
// All Rights Reserved.
//=====================================================================
IBMCCSID=697 // IBM Character set ID
BMCodePage=37 // IBM codepage
IBMAS400Keyboard=CAI // AS/400 Keyboard type
WindowsFontCharSet=0 // (ANSI)
```

If, for example, you want to create a custom code page that has Canadian French settings except for the keyboard type, you could change the keyboard type from CAI to USB and then save it as custom.codepage.

`BMAS400Keyboard=USB // AS/400 Keyboard type (custom)`

The lower portion of the code page sample is a series of character mappings. A character mapping consists of 3 values on a single line, separated by spaces or commas:

Host (EBCDIC) value, prefixed with H. They can be specified as decimal or hexadecimal numbers.

Windows single-byte value used to display a given character in the Design Tool, prefixed with W. They can be specified as character literals, decimal values, or hexadecimal values.

6 bit Unicode character, prefixed with U. They can be specified as character literals when they are 8 bits or less, decimal, or hexadecimal values.

In the samples, mappings that are the same as the US English default are commented out with // at the beginning of the line. For example, the French.codepage.sample has H44 mapped to W '@' and U '@' for the 'commercial at' sign and is not commented out. (In US English, H44 is mapped to lowercase a with a grave accent sign.) You could create a custom code page based on the French code page, with a modification that returns H44 to the US English default.

Using the Custom Code Page Log File

After creating and saving the code page and then restarting the Host Integrator server or Design Tool, look at the log file for the code page: `<Host Integrator install folder>\codepage\codepage.log`.

When the code page is deployed successfully, the log looks like this:

```
---------------------------------------------------------------------------
    Custom codepage logfile opened: Thu Jun 22 08:17:18 2006
---------------------------------------------------------------------------



Info Duplicate character mapping: We8 ==> U0e48
Info Duplicate character mapping: We9 ==> U0e49
Info Duplicate character mapping: Wea ==> U0e4a
Info Duplicate character mapping: Web ==> U0e4b
Info Duplicate character mapping: Wec ==> U0e4c
Info === Read 221 Lines of text
Info === Found 99 host character mappings
Info +Added Mapping: H41 <==> Wa0
 …
 …
 Info === Found 94 unicode character mappings
 Info +Added Mapping: : W1d <==> U00a2
 Info +Added Mapping: : W1e <==> U00a6

 ---------------------------------------------------------------------------
 Custom codepage logfile closed: Thu Jun 22 08:17:18 2006
 ---------------------------------------------------------------------------
```

When a code page has errors, the log file includes warnings such as the ones shown below:

```
Warn Line 11 Unrecognized property "MyProperty"
Warn Line 20 !Host value invalid or out-of-range.
Warn Line 20 !Windows value invalid or out-of-range
Warn Line 20 !Unicode value invalid or out-of-range.
Warn Line 20 !Character mapping ignored "H3F U10000"
Warn Line 22 !Host value invalid or out-of-range.
Warn Line 22 !Windows value invalid or out-of-range
Warn Line 22 !Unicode value invalid or out-of-range.
Warn Line 22 !Character mapping ignored "HH WW UU"
```

4.4.3 Mapping the Keyboard

Mapping the Keyboard

You can customize your keyboard to associate keystrokes with Design Tool commands or terminal keystrokes. This process is known as keyboard mapping.

A keystroke can be:

A single key, such as K, F1, or Num Lock

A combination of keys that you press at the same time, such as Ctrl+F2 or Alt+Shift+M

When you use more than one key in a keystroke, all keys preceding the final key must be modifier keys—Alt, Ctrl, or Shift. You can create keystrokes with a single modifier key (Ctrl+F7) or with multiple modifier keys (Shift+Ctrl+F7).

The command that you map to a keystroke can be:

A terminal key, such as Attention or PF1

A Design Tool command, such as .aboutDlg or .Connect

Any combination of the above—you can build the Command string to include multiple terminal keys, and commands.

Drag-and-Drop Keymap Options

The following are some shortcuts you can use in the Keyboard Setup dialog box:

- Click a PC keystroke and then, holding down the left mouse button, drag it to the terminal keyboard. Already mapped keys become small cyan rectangles; unmapped keys become a rectangle, but they don't change color. When you move this rectangle over a key on the terminal keyboard, a black outline appears around this key (indicating correct positioning over the key).

The rectangular mouse cursor either retains its form (indicating that you can map to this key) or turns into a red circle with a line through it (indicating that you cannot map to this key).

- The same process works in reverse. Select a terminal keystroke, and then drag it to the PC keyboard.
- To remove mapping from a key, click the key, and then drag it off the Keyboard Setup dialog box. When you drag the key off the edge of the dialog box, the image of the key turns into a trash can; you can then release the left mouse button and the mapping is cleared.

Changing the PC or Terminal Keyboard

The Keyboard Type dialog box lets you specify which PC and Terminal keyboard are shown in the Keyboard Setup dialog box. On the Settings menu, click Keyboard to open the Keyboard Setup dialog box, and then click Keyboards to open the Keyboard Type dialog box. The Keyboard Type dialog box contains two list boxes:

PC - Click the down arrow to see the available PC keyboards.

Terminal - Click the down arrow to see the available terminal keyboards.

Other options that affect keyboard mapping are available in the View Settings dialog box (available from the Settings menu).

Identifying a Keystroke

To identify a keystroke:

1. Click any modifier keys (Alt, Ctrl, or Shift) that you want to use in the keystroke. This step is optional. You can create keystrokes with no modifier keys (such as F7), with a single modifier key (Ctrl+F7), or with multiple modifier keys (Shift+Ctrl+F7).
2. Select a primary key.

This can be any key on the PC keyboard--except an unavailable key. Cyan keys are already mapped. To remap a key, click Remove, and then continue.

If your keyboard keys on your keyboard do not exactly match the PC keyboard keys shown in the Keyboard Setup dialog box, use your keyboard, instead of the mouse, to select a PC keystroke. That way, you'll be able to see how your keyboard keys correspond to the keys on the Design Tool's graphical PC keyboard. Select a key, click an empty spot anywhere under **PC Keyboard**, and use your keyboard to press the keys you want. Do not use your mouse to click any keys.

Note

You cannot remap the Windows logo function keys.

Removing a Keystroke's Mapping

Select the keystroke on the PC keyboard in the Keyboard Setup dialog box

Click the Remove button.

The key's mapping is cleared (so that pressing the keystroke has no effect in the Design Tool).

Resetting a Keystroke's Mapping

To reset a keystroke to its default setting...

Identify a PC keystroke in the Keyboard Setup dialog box.

Click the Default button.

Don't confuse the Default button with the Defaults button (which deletes all keyboard customizations).

This procedure only works for PC keystrokes mapped to an action other than the default action for the keystroke.

Restoring the Default Keyboard Mapping

To restore the default keyboard mapping, click Defaults in the Keyboard Setup dialog box.

All keyboard customizations are removed. For information on the default keyboard mapping for VT and HP terminals, click a mapped cyan/teal PC key and note the corresponding Action in the lower half of the Keyboard Mapping dialog box.

Determining Keyboard Mapping

Shading, color, and key appearance are used in the Keyboard Setup dialog box to show how your keyboard is mapped. Cyan keys are mapped, by default, but their mappings can be changed.

Keys not appearing in cyan are not mapped. (Letter and number keys are not considered mapped because it is unlikely you would want to redefine the mappings for these keys.)

If you want to see how a key on the PC is mapped, click the key to select it; the key is selected. If this PC key is mapped to a terminal key, the corresponding key on the terminal keyboard, below, is also selected. For example, click the F1 key on the PC keyboard. It changes color, as does the F1 key (in 5250 sessions) or the PF1 key (in 3270 sessions) on the terminal keyboard. Click F1 on the PC keyboard again to clear the key.

Note

Num Lock key status affects the mapping: If you click the Num Lock key while the Keyboard Setup dialog box is open, you must exit and return to the dialog box before that change is reflected in the displayed keyboard mapping.

Some keys are mapped to Design Tool commands. For example, click the Caps Lock key on the PC keyboard. The terminal keyboard is replaced by a set of fields.

Caps Lock is mapped to the Design Tool command `.Toggle(rc_CapsLockState)`, which turns capital letters on or off.

Click Caps Lock again to clear it.

If you click a modifier key (Alt, Ctrl, or Shift) on the PC keyboard, the color pattern on the remaining keys changes. For example, try clicking Alt. The keys that turn cyan are mapped to Alt. For example, in 5250 sessions, the F1 and D keys become cyan when you click Alt. This means that Alt+F1 and Alt+D are mapped in the Design Tool's default keyboard mapping.

Keys that appear dimmed when you click Alt, Ctrl, or Shift are currently mapped (in combination with the selected modifier key) and cannot be changed. Typically, these are keystrokes that are defined by Windows. For example:

Keystroke	Map to setting
Alt+F4	Close the current window (Windows standard)
Ctrl+Esc	Display the Windows task list (Windows standard)

Linking a Keystroke to a Command

Once you have identified a PC keystroke and one or more commands to map it to, click the Map button.

Mapping a Keystroke

There are three things you need to do in the Keyboard Setup dialog box to map a keystroke:

Identify the keystroke

Identify one or more commands

Link the keystroke to the commands

With the Design Tool's Keyboard Setup dialog box, you can identify either the keystroke or the action first, but the linking process must be the final step.

Keyboard settings can be saved to a settings file (`.dtool`) with other configuration information.

Mapping a Keystroke

There are three things you need to do in the Keyboard Setup dialog box to map a keystroke:

Identify a keystroke

Identify one or more commands

Link the keystroke to the command

With the Design Tool's Keyboard Setup dialog box, you can identify either the keystroke or the action first, but the linking process must be the final step. Keyboard settings can be saved to a settings file (.dtool) with other configuration information.

To identify a keystroke

1. Click any modifier keys (Alt, Ctrl, or Shift) that you want to use in the keystroke.

   This step is optional. You can create keystrokes with no modifier keys (such as F7), with a single modifier key (Ctrl+F7), or with multiple modifier keys (Shift+Ctrl+F7).

2. Select a primary key.

   This can be any key on the PC keyboard--except an unavailable key. Cyan keys are already mapped. To remap a key, click Remove, and then continue.

   If your keyboard keys on your keyboard do not exactly match the PC keyboard keys shown in the Keyboard Setup dialog box, use your keyboard, instead of the mouse, to select a PC keystroke. That way, you'll be able to see how your keyboard keys correspond to the keys on the Design Tool's graphical PC keyboard. Select a key, click an empty spot anywhere under **PC Keyboard**, and use your keyboard to press the keys you want. Do not use your mouse to click any keys.

Note

You cannot remap the Windows logo function keys.

Removing a Keystroke's Mapping

Select the keystroke on the PC keyboard in the Keyboard Setup dialog box

Click the Remove button.

The key's mapping is cleared (so that pressing the keystroke has no effect in the Design Tool).

Resetting a Keystroke's Mapping

To reset a keystroke to its default setting:

Identify the keystroke in the Keyboard Setup dialog box.

Click the Default button.

Tip

Don't confuse the Default button with the Defaults button (which deletes all keyboard customizations).

This procedure only works for PC keystrokes mapped to an action other than the default action for the keystroke.

Restoring the Default Keyboard Mapping

To restore the default keyboard mapping, click Defaults in the Keyboard Setup dialog box.

All keyboard customizations are removed. For information on the default keyboard mapping for VT and HP terminals, click a mapped cyan/teal PC key and note the corresponding Action in the lower half of the Keyboard Mapping dialog box.

Determining Keyboard Mapping

Shading, color, and key appearance are used in the Keyboard Setup dialog box to show how your keyboard is mapped. Cyan keys are mapped, by default, but their mappings can be changed.

Keys not appearing in cyan are not mapped. (Letter and number keys are not considered mapped because it is unlikely you would want to redefine the mappings for these keys.)

If you want to see how a key on the PC is mapped, click the key to select it; the key is selected. If this PC key is mapped to a terminal key, the corresponding key on the terminal keyboard, below, is also selected. For example, click the F1 key on the PC keyboard. It changes color, as does the F1 key (in 5250 sessions) or the PF1 key (in 3270 sessions) on the terminal keyboard. Click F1 on the PC keyboard again to clear the key.

Note

Num Lock key status affects the mapping: If you click the Num Lock key while the Keyboard Setup dialog box is open, you must exit and return to the dialog box before that change is reflected in the displayed keyboard mapping.

Some keys are mapped to Design Tool commands. For example, click the Caps Lock key on the PC keyboard. The terminal keyboard is replaced by a set of fields.

Caps Lock is mapped to the Design Tool command `.Toggle(rc_CapsLockState)`, which turns capital letters on or off.

Click Caps Lock again to clear it.

If you click a modifier key (Alt, Ctrl, or Shift) on the PC keyboard, the color pattern on the remaining keys changes. For example, try clicking Alt. The keys that turn cyan are mapped to Alt. For example, in 5250 sessions, the F1 and D keys become cyan when you click Alt. This means that Alt+F1 and Alt+D are mapped in the Design Tool's default keyboard mapping.

Keys that appear dimmed when you click Alt, Ctrl, or Shift are currently mapped (in combination with the selected modifier key) and cannot be changed. Typically, these are keystrokes that are defined by Windows. For example:

Keystroke	Map to setting
Alt+F4	Close the current window (Windows standard)
Ctrl+Esc	Display the Windows task list (Windows standard)
Ctrl+Alt+Del	Restart system (DOS standard)

Linking a Keystroke to a Command

Once you have identified a PC keystroke and one or more commands to map it to, click the Map button.

4.4.4 Setting Colors and Fonts

Configuring Colors

The Colors tab in the Display Setup dialog box lets you associate colors with screen attributes in the Terminal window.

Color settings can be saved to a settings file (`.dtool`) with other configuration information.

Setting Colors with a Mouse

While the Colors tab is displayed, move your cursor outside of it. The cursor now looks like a pointing hand.

Place the cursor over text in the Terminal window and click the left button. In the Colors tab, the **Item** box shows the attribute of the text you clicked.

Move the cursor back into the dialog box and click a color in the **Foreground (text)** box. All text with the attribute shown in the **Item** box now appears in the new color.

Alternatively, you can click a color first and then, holding down the left mouse button, drag the cursor off the Colors tab. In this case, the cursor becomes a paint brush.

As you move the paint brush cursor across the Terminal window, the value in the **Item** box changes to show the attribute of the text beneath the paint brush.

When the cursor is over an area whose color the Design Tool cannot change, the paint brush cursor looks like a paint brush with a slashed circle covering it.

When you release the mouse button on text with an appropriate attribute, the text--and any additional text that shares this attribute--are set to the specified color.

Click Defaults if you want to reset all colors to their default values. Click Save on the File menu to retain the new color settings after you exit the Design Tool.

Setting Colors for an Attribute

From the Settings menu, select Display... to open the Display Setup dialog box.

In the Item box, select a value. The Sample Text window shows the current colors for the selected item--whether or not any part of your current Terminal window actually has the same attribute.

In the Foreground (text) box, select a color. Any text matching the selected attribute immediately changes color as you move from color to color.

You can return to the Item box and set a color for a different attribute, or click OK to exit the Display Setup dialog box.

Click Defaults to reset all colors to their default values.

Configuring Display Fonts

Use this dialog box to specify the display font and style in the Design Tool.

Changing the Design Tool's font has no effect on the font used by your printer, the status line, menu commands, or dialog boxes.

Font

Specifies the display font in the Design Tool session window. You can use any font, including TrueType fonts. Arial is an example of a TrueType font that is not monospaced, and therefore is not supported by the Design Tool.

By default, the Design Tool uses the r_ansi font. This provides a 24 x 80 display that accurately emulates the terminal.

When you resize the Terminal window, the Design Tool chooses a new font size so the correct number of rows are displayed on the screen. If the Design Tool cannot display the font you have chosen, the default (r_ansi) is displayed instead.

The available fonts can change as you change the model ID in the Session Setup dialog box, because not all model IDs support all fonts.

Note

When you print all or part of a host screen from a terminal session, the font used is the currently configured display font.

Font Style

Specifies a font style, regular or bold. The Design Tool does not support italicized fonts. The default is Regular.

Display variable width fonts

Specifies whether proportionally-spaced font types appear in the Fonts box. The default is Yes.

Auto sizing

Specifies that for whichever font or style you select, the Design Tool automatically adjusts the font size to fit all text in the Terminal window. To change the font size, set this setting to No. By default, Auto Font Size is enabled.

Fonts Installed by the Host Integrator

The following files are installed automatically by the Host Integrator installation program:

Font Type	Description
Dialog.fon	The font used in the Design Tool's dialog boxes.
_ansi.ttf	The file for the TrueType versions of the r_ansi fonts.
R_ansi.fon	The file for the bitmap versions of the r_ansi fonts.
R_apl.ttf	The file for the TrueType versions of the r_apl fonts. You can't select this font from the Fonts tab—it's used to display characters in the Operator Information area (OIA) only.
R_contr.fon	You can't select this font from the Fonts tab—it's used to display control characters in VT and HP terminal emulation sessions.
R_contr.ttf	You can't select this font from the Fonts tab—it's used to display control characters in VT and HP terminal emulation sessions.
R_ibmasc.fon	The r_ibmasc font.
R_ibmhex.fon	You can't select this font from the Fonts tab—it's used only when you're in hexadecimal mode.

Font Type	Description
R_ibmsl.fon	The font used to display host symbols and status lines.
R_symbo_.fon	The file for the symbol fonts.
R_symbol_.ttf	The file for the TrueType versions of the symbol fonts.

To use a monospaced (fixed-pitch) font other than Host Integrator's default display font, r_ansi, that font must already be installed under Windows. If the font you want to use doesn't appear in the Font box in the Display Setup dialog box, you need to install it using Windows Control Panel. Refer to your Windows documentation for information on installing new fonts.

4.4.5 Command List Edit

Use the Command List Edit dialog box to create a login, a logout, or a move cursor command list to be saved in the model file. To view a detailed example of a login and logout command list, see the Pine model example.

Command lists will be generated if operation generation is on (this is the default).

Configure your login or logout command lists to be automatically executed on a host connect or disconnect by selecting the Execute login command list and Execute logout command list check boxes on the General tab of the Preferences Setup dialog box.

Select a command from the Command list and press the F1 key for online help on that command.

Use the Copy button to grab an image of the whole command list, which you can them paste into a text editor making the entire list visible.

To create a login command list

On the Connection menu, select Session Setup to open the Session Setup dialog box.

In the Session box, select the terminal type from the Type box.

Type the host name or IP address in the Host name or IP address box and click Connect.

On the Model menu, point to Record and click Start Recording. Type your username and password and advance to the next screen.

On the Model menu, point to Record and click Stop Recording.

On the Stop Recording dialog box, select Save as login command list and click Save to open the Command List Edit dialog box.

View and edit the commands associated with the login command list that you've just created in the Commands box and click OK to save it.

To view and edit the login command list, use the buttons next to the Login (after host connect) box in the Model Properties dialog box.

To execute your login command list, disconnect and then reconnect to the host, and click the Execute login command list button on the standard toolbar. Alternatively, you can open the Model Properties dialog box and click the Execute button.

To create a logout command list

Move to the home entity of the model.

When you're ready to logout of the host, click Start Recording.

Type the command used to logout of the host on the terminal screen.

On the Stop Recording dialog box, select Save as logout command list and click Save to open the Command List Edit dialog box. By default, the Stop Recording dialog box opens if you are disconnecting from a host while recording.

View and edit the commands associated with the logout command list that you've just created in the Commands box and click OK to save it. Delete the .MoveCursor command.

To view or edit the logout command list, use the buttons next to the Logout (before host disconnect) box in the Model Properties dialog box. Make sure to save your model before exiting.

To execute your logout command list, disconnect and then reconnect to the host, and click the Execute logout command list button on the standard toolbar. Alternatively, you can open the Model Properties dialog box and click the Execute button.

Notes:

- It is recommended that you create a logout command list on your home entity. If you record a logout command list from an entity other than the home entity, it may not return the expected results when run in either the Design Tool or on the server. This happens because the default behavior of the logout command is to navigate to the home entity and then execute the logout command list. If you are not on the home entity when recording the logout command list, the following message will appear:
  `"Execution of the logout command list will always start at the home entity. Since recording started at a different entity, we will add a 'Navigate to <name> entity' command to the start of your command list. At run-time, this may cause unnecessary host navigation. Do you want to continue?"`
- If you create a model that uses a login command list that goes past at least one entity to reach the home screen entity, it is not necessary to exit the model using the same sequence. For example, a login command list may include a sequence to reach the home entity using signon entity "A," main menu "B," and home entity "C." The appropriate logout command list should, however, should proceed directly from home entity "C" to signon entity "A."

To create a move forward or move backward command list

1. Click the right arrow button and select the terminal key you want to use to move the cursor forward.

   Select Tab from the Key list and then configure your tab position settings using the Cursor tab on the Entity window.
2. Click OK to save your move forward command list and return to the Advanced Model Properties dialog box.

Follow the same directions to create a move backward command list.

Build Command

The Build Command dialog box provides the following information about the selected command:

Description - displays a read-only description

Syntax - provides a read-only text description

Parameters - contains the values that determine the characteristics or the behavior of the command

4.5 Importing Model Elements

Elements of an existing model can be imported into other models. Developers can work simultaneously on the same model, importing the different elements to create a new model. You can also reuse parts of a model, streamlining your work and letting multiple developers work efficiently.

Settings and properties are not inherited from the source model. You can reconfigure them in the destination model if you want to keep the existing settings and properties.

Terminology and Best Practices for Importing Models

Copy Objects

Copy Entity

4.5.1 Workflow

1. Open the destination model in the Design Tool.

   The destination model is the model that you are importing to. It is a good idea to always backup your model before importing model elements.

2. From the File menu, choose **Import Model Elements** to open the Import Model Elements dialog box. You can open either a `.modelx file` or `a .model file`.

3. Browse to the import model. You import from the source model (in the dialog box) to the destination model, which is open in the Design Tool.

   The left pane displays a tree representation of the source model, including all entities, tables, and variables.

4. Choose the elements of the model you want to import into the destination model.

> **Note**
>
> Model elements are those items that can be imported and comprise entities, attributes, operations, recordsets, recordset fields, tables, table columns, procedures, compound procedures, and model variables. When an operation containing a `WaitForMultipleEvents` command is imported, the referenced Host Events are also imported.
>
> Patterns and events cannot be imported separately from their entity.

5. Preview your selected elements in the Destination pane. This pane is divided into the following elements.

   Insert elements are new elements that are added to the model when the import is complete.

   Update elements are elements that are updated after the import is complete.

   Referenced elements are elements that are referenced by other elements. If these are not resolved, either manually or automatically, a temporary replacement version of the element is created. These replacement elements are marked with a _notImported suffix.

6. Click OK.

> **Note**
>
> If an element already exists in the destination model, importing it overwrites the current element.
>
> By default, the destination model with all imported model elements is validated using the Host Integrator Validator.

7. Resolve any validation errors.

8. If you are importing a model that contains an event handler, copy the Java code from the model source Scripts sub-directory to the corresponding sub-directory in the destination model.

A reference to the event handler is imported along with the parent element.

4.5.2 Terminology and Best Practices for Importing Models

Elements of an existing model can be imported into other models. Developers can work simultaneously on the same model, importing the different elements to create a new model. You can also reuse parts of a model, streamlining your work and letting multiple developers work efficiently.

Terminology

- Destination model

  The destination model is the model that you are importing to and the model which is open in the Design Tool.

- Master model

  After the import is complete, the new model that was created by importing elements from the source model into the destination model is called the master model.

- Source model

  You import from the source model to the destination model. The source model is selected in the Model Import dialog box.

- Model elements

  Model elements are those items that may be imported and comprise entities, attributes, operations, recordsets, recordset fields, tables, table columns, procedures, compound procedures, and model variables.

- Referenced elements

  A referenced element is referenced by another element, but not selected for import. All referenced elements must be resolved, either manually or automatically, before the new model can be deployed.

Note

If the option **Automatically select referenced elements** is selected on the Preferences dialog box, then `_notImported` elements do not occur. If this option is cleared, then these replacement elements must be resolved manually.

Best Practices

- Always backup your model before you begin the import process.
- Understand your model.
- Elements that are identified as `_notImported` in the preview pane of the Model Import dialog box are not problems. It is not unusual to have elements that you do not want to import.

  An element, for example an attribute, might reference another element, such as a variable. When the attribute is imported, but the variable isn't, then the referenced element is missing. The import function creates a temporary replacement version of the referenced element. These replacements are marked with the `_notImported` suffix and inserted into the destination model.

  Since all referenced elements must eventually be resolved before a model can be deployed, after finishing the import, use the Design Tool to modify the attribute you have just imported. Find the reference to the placeholder variable and change it to refer to a "real" variable in your model.
- Create a base model with all the entities up to the home entity. This model should include a model of lifecycle event handlers.

  If all the screens you want to model are known in advance, it is beneficial to include the entities that will be duplicated across different models in the base model. This prevents a particular entity from having different names across different models and reduces the size of the model.
- Developers should use the base model and extend it to include other model elements. The model should be given a unique name. For example, `<DeveloperInitials>` `_<BaseModelName>`.
- Add a prefix to the names of event handlers to make them unique to each model. This ensures that there are no clashes during the import process.
- Code freeze both models during the import merge process.

Test the destination (master) model when the import is complete.

Provide the master model to the developers to continue the task of extending the model.

Frequently Asked Questions

- Can I import models with event handlers?

  Yes. If you import a model that contains an event handler, a reference to it is imported along with the parent element. However, you must manually copy the Java code from the model source Scripts sub-directory to the corresponding sub-directory in the destination model files.
- How do I make sure I don't overwrite existing elements during import?

  When you select an element that is already in the destination model the element is replaced. However, you must confirm your decision before the element is overwritten. This option is set on the **Import** tab of the Preferences Setup dialog box and is selected by default.

- Do I have to validate my new master model?

  By default, all import models are validated in the Host Integrator Validator. A model must successfully validate before it can be deployed.

- Can I change how sub-items and referenced items are selected in the Model Import pane?

  Options that determine how to work with the Import Model Elements dialog box are set on **Settings** > **Preferences** > **Import**. See Import Preferences for a description of each option, as well as the default settings.

- What do the = green and = yellow equal signs mean?

  All of the element's properties in the source model are the same as those in the destination model. For example, if an entity exists in both models and all of the entity's sub-items are also marked with a = green equal sign, you do not have to import this element.

  However, a = yellow equal sign indicates that the element properties are the same, but there are differences between one or more sub-items of the element.

More information

Creating a model

4.5.3 Copy Objects

You can copy objects you have already created for reuse. Copying patterns, attributes, or recordsets allows you to quickly create new model objects based on objects that you have already defined within a given entity.

You can copy patterns, attributes, and recordsets as described below. You can also copy tables and procedures, and complete entities can be replicated a shown in Copy Entity.

1. From the drop down list attached to the following buttons, click Create Copy:
   - Attribute
   - Recordset
   - Pattern
   - Operation

   The copied object has a similar name to the original object with a number appended (for example, Password_2).

2. Make the necessary modifications to the new copy.

Avoiding Problems

Use these best practices to avoid common problems:

If you copy a pattern that is configured to be a signature pattern for an entity, edit the pattern characteristics to avoid invalid signatures.

When you copy an entity, you must identify a signature that uniquely identifies the entity. It cannot have the same signature as the source entity you copied from.

Avoid copying an object that is relative to a pattern, since the relative relationship is usually not the same for the new object.

Copying or importing operations that act on entity-specific objects may not function as originally designed, even if each command in the operation is copied successfully.

4.5.4 Copy Entity

You can copy entities you have already created. This is a fast way to build models that contain multiple occurrences of similar screens.

To Copy an Entity

1. Click **Copy From** on the options list next to the **Entity** button. This option is available when you are on an undefined screen.
2. Select the entity to copy.

> **Note**
>
> On the left side of the Copy Entity object dialog box is a list of entities within this model. Entities that cannot be copied are marked with a red X.

3. Confirm that this item can be copied as you expect.

   The right pane's Snapshot tab includes a snapshot of the selected entity. The Entity tab shows the name of the source entity and the objects within the entity.

   The **Copy status** column indicates whether the entity can be copied. If an object cannot be copied (for example, you are trying to copy an attribute from a 132 column entity to an 80 column entity), an error icon is displayed.
4. Click **Copy**.
5. Edit the resulting entity.

   At a minimum, you must identify a signature that uniquely identifies the entity. It cannot have the same signature as the source entity you copied from.

Modify any patterns that are defined for the source entity and are not present in the target entity. They are listed with a red X to the left of the pattern name.

More information

Adding Entities to a Model

4.6 Adding Entities to a Model

An entity is a unique host application screen. After you label your host application screen as an entity, you must define it using entity definitions and entity properties.

Note

Object names for entities, attributes, operations, recordsets, recordset fields, and, variables are not case sensitive.

To add an entity to a model:

1. When the first screen appears after connecting to the host, click the **New Entity** button in the Entity window. (All of the other options are unavailable at this point.)
2. By default, the entity is named **Entity_1**. To change the name, select it and type a new name in the Entity box.
3. Click **Apply** to save your changes.
4. Now, add a pattern to the entity. See Adding Patterns.

You can also copy entities as shown in Copy Entity.

4.6.1 Adding Patterns

A pattern is a selected area on an entity that does not contain data that changes from session to session.

To add a pattern to an entity:

1. After you've created an entity, use the cursor to select a static section of the host screen to create a pattern.

   Each pattern can have up to 259 characters.

Tip

You can choose to auto-generate patterns by clicking the **Auto-generate** button, but you will still have to assign properties and error conditions to these patterns on the Pattern tab.

2. On the Pattern tab, click the **New Pattern** button. All of the other options on this tab are unavailable at this point.
3. By default, the pattern is named **Pattern_1**. To change the name, select it and type a new name in the **Name** box. The Design Tool has now recorded the position, properties, and the signature parameters of the pattern.
4. Click **Apply** to save your changes.
5. Now, add attributes to identify text fields on the entity as shown in Adding Attributes.

To copy a pattern:

After you have created a pattern, you may need to create another that is similar. You can copy the pattern and then make the necessary modifications to the copy.

Select the pattern you want to copy.

Click the list next to the **New Pattern** button and then select **Create Copy**. A new operation with a similar name and a number appended to the end is created. For example: Pattern_2_2.

Modify the new pattern so that it is distinguishable from the pattern on which it is based.

Tip

If you copy a pattern that is configured to be a signature pattern for an entity, be sure to edit the pattern characteristics to avoid invalid signatures.

4.6.2 Adding Attributes

An attribute is a selected area on an entity containing data that needs to be accessible via the model file.

After you've created an entity and added pattern to that entity, you're ready to add an attribute or attributes.

Tip

You can choose to auto-generate attributes by clicking the Auto-generate button, but you will still have to assign variables, properties, and error conditions to these attributes on the Attribute tab.

To add an attribute to an entity:

1. Use the cursor to select a text field on the host screen.
2. On the Attribute tab, click the **New Attribute** button. All of the other options on this tab will be unavailable.
3. By default, the attribute is named **Attribute_1**. To change the name, select it and type a new name in the **Name** box.

   The Design Tool records the position and other defining characteristics of the attribute.

   You can accept the Design Tool's default properties, or change them on the Attribute tab.
4. Click **Apply** to save your changes.

Some entities require that you add recordset information. After adding these elements to an entity, you can define operations to navigate between entities.

To copy an attribute:

After you create an attribute, you may need to create another that is similar. You can copy the attribute and then make the necessary modifications to the copy.

Select the attribute you want to copy.

Click the list next to the **New Attribute** button and select **Create Copy**. A new attribute with a similar name and a number appended to the end is created (for example, `Password_2`).

Modify the new attribute so that it is distinguishable from the attribute on which it is based.

To add colors to an attribute:

1. From the **Settings** menu, select **Display**... to open the **Display Setup** dialog box.
2. In the **Item** box, select a value.

   The Sample Text window shows the current colors for the selected item and whether any part of your current Terminal window actually has the same attribute.
3. In the **Foreground (text)** box, select a color.

   Any text matching the selected attribute immediately changes color as you move from color to color.
4. To set a color for a different attribute, return to the **Item** box.

5. Click **OK** to exit the Display Setup dialog box.
6. Click **Defaults** to reset all colors to their default values.

More information

Customizing the Terminal

4.6.3 Adding Recordsets to an Entity

To add a recordset to an entity:

1. On the Recordset tab, select a scrollable area on the terminal screen and click the **New Recordset** button in the **Name** box.
2. By default, the recordset is named **Recordset_1**. To change the name, select it and type a new name in the **Name** box.
3. Configure your recordset using the **Recordset Position**, **Layout**, and **Fields** tabs.
4. Click **Apply** to save your changes.

To copy a recordset:

After you have created a recordset, you may need to create another that is similar. You can copy the recordset and then make the necessary modifications to the copy.

Select the recordset you want to copy.

Click the list next to the **New Recordset** button and select **Create Copy**. A new recordset with a similar name and a number appended to the end is created (for example, `AcctTransData_2`.

Modify the new recordset so that it is distinguishable from the recordset on which it is based.

Testing Recordsets

This is an example of testing a fetch record in the CICSAccts model.

On the File menu, choose **Open** and select `CICSAccts.modelx`.

Click **Return** to connect.

From the **Entity** list on the Entity window, select **NameSearchResults**.

Select the **Recordset** tab and click the **Test** button in the **Name** box.

In the Test Recordset dialog box, select **Fetch Records** from the **Action** box.

Click the **Execute** button.

View the results of the data fetch test in the **Fetch returned** box.

Fetch Records

To test a record fetch:

Open **CCSDemo.modelx** in the Design Tool.

On the Connection menu, choose **Connect** to localhost via Telnet.

From the **Entity** list on the Entity window, select **NameSearchScreen**.

Select the **Recordset** tab and click the **Test** button in the **Name** box.

In the Test Recordset dialog box, select **Fetch Records** from the **Action** box.

To filter out a record or records from within a recordset, click the **Edit** button to open the Filter String Edit dialog box.

Click the **Execute** button.

View the results of the data fetch test in the **Fetch returned** box.

> **Note**
>
> After executing the first **Fetch Records**, you must reset the recordset by executing the **Set Current Record Index** action.

Inserting a Record

To insert a specific record or multiple records into a recordset:

Confirm that SIDemo is running in the Host Emulator, then open `SIDemo.modelx` in the Design Tool.

On the **Connection** menu, click connect to localhost via Telnet.

From the **Entity** list on the Entity window, select **CustomerPurchases.**

Click the Recordset tab and select **CustomerList** from the **Name** box.

Click the Test button to open the Test Recordset dialog box.

Select **Insert Record** from the **Action** box.

In the **Insert record** box, notice that the **Select** and **Customer** fields have been created for this recordset and are listed in rows.

To insert a record, under the **Value** column, type `B. JONES` into the **Customer** text box.

Click **Execute**. If the insert is successful, the record contents should appear in the Terminal window.

Note

After you execute an insert on a recordset, you need to reset the recordset by either navigating away from it and returning, or executing the home operation on the recordset before executing any other method on it.

Selecting a Record

There are two possible ways to select a record: by condition or by index. The following procedures use `SIDemo.modelx` as an example.

To select a specific record by condition in a recordset:

Confirm that SIDemo is running in the Host Emulator, then open `SIDemo.modelx` in the Design Tool.

On the Connection menu, click Connect to localhost via Telnet. Confirm that SIDemo is running in the Host Emulator, then open `SIDemo.modelx` in the Design Tool.

In the **Entity** box, select **CustomerPurchases** and click the Recordset tab.

Click the **Test** button to open the **Test Recordset** dialog box.

From the **Action** box, select **Select Record**.

Click the **Edit** button to open the Filter String Edit dialog box.

In the Filter String Edit dialog box, select **Customer** from the **Recordset fields** box, click the = button, and type the following in the **Filter string** box: `Q. ARMSTRONG` The filter string should appear as follows: `CustomerList.Customer = "Q. ARMSTRONG"`

Click **OK**.

In the **Test Recordset** dialog box, click **Execute**.

If the selection is successful, the record contents for `Q. ARMSTRONG` are displayed on the terminal screen along with the following message:

`The selection operation reached an expected destination.`

Note

This action is the analog of the SelectRecordByFilter method.

If the record exists on an entity that does not contain any defined recordsets, the Test Recordset dialog box closes.

To select a specific record by index in a recordset:

This action is the analog of the `SelectRecordByIndex` method. If the record exists on an entity that does not contain any defined recordsets, the Test Recordset dialog box closeS.

Follow the procedures described in Step 1 through Step 4 above.

From the **Action** box, select Set **Current Record Index**.

In the **Set the current record index** to box, type `3`, and click **Execute**. To check that your current record index is set to the line number you want, view the indicator (for example, **Current record index:** `<line number>` at the bottom of this dialog box.

Select **Select Record** from the **Action** box and click **Execute**.

If the select is successful, the record contents for `Q. ARMSTRONG` is displayed on the terminal screen along with the following message:

```
The selection operation reached an expected destination.
```

Update Current Record

To update the current record in a recordset:

Open **Purchases.modelx** in the Design Tool.

On the Connection menu, choose **Connect to localhost via Telnet**.

In the **Entity** box, select **CustomerPurchases** and click the Recordset tab.

Click the **Test** button to open the Test Recordset dialog box.

From the **Action** box, select **Set Current Record Index**.

In the **Set the current record index to** box, type `2`.

Click **Execute**. To check that your current record index is set to the line number you want, view the indicator (for example, **Current record index:** `<line number>`) at the bottom of this dialog box.

From the **Action** box, select **Update Current Record** and click **Execute**. Look in the **Update current record** box and notice that Select appears in the **Field** column and the **Value** column is blank.

9. In the **Value** column, Type `1` and then click **Execute**.

   The value you entered (`1`), is displayed on line 3 of the **Select** column in the Terminal window.

Tip

To test any scrolling operations that have been defined, click any of the available buttons in the **Perform Scrolling Operation** box. If there are no scrolling operations in the recordset, the buttons are unavailable.

Set Current Record Index

To set the current record index of a recordset:

Confirm that SIDemo is running in the Host Emulator.

In the Design Tool, open `SIDemo.modelx`.

On the Connection menu, click **Connect to localhost via Telnet**.

In the **Entity** box, select **CustomerPurchases** and click the Recordset tab.

Click the **Test** button to open the Test Recordset dialog box.

From the **Action** box, select **Set Current Record Index**.

In the **Set the current record index to** box, type `2`.

Click **Execute**.

To check that your current record index is set to the line number you want, view the indicator (for example, **Current record index:** `<line number>`) at the bottom of this dialog box.

From the **Action** box, select **Fetch Records** and click **Execute**. Look in the **Fetch returned** box and notice that records were fetched beginning with line 3: `Q. ARMSTRONG` appears in the **Customer** column.

Note

To test any scrolling operations that have been defined, click any of the available buttons in the **Perform Scrolling Operation** box.

4.6.4 Defining Entities and Recordsets for Procedures

Host Integrator fulfills SQL queries from client applications by navigating to the pertinent entities in a host application that contain the table data and either reading, modifying, or deleting the data. A procedure's definition must specify all entities and recordsets that contain table data.

In addition to defining the entities and recordsets that contain table data, you can also define branching entities. These entities provide some flexibility for a procedure when it is traversing through a host application, as well as error entities that you can use to trap errors in a procedure.

- Adding Entities to a Procedure

  You must add every entity to a procedure that contains the attributes and/or recordset field that the procedure is able to query.

- Adding Recordsets to a Procedure

Recordsets are areas on an entity that contain dynamically changing information, usually scrolling sets of data that are a result of a data fetch.

- Adding Branch Entities to a Procedure

  Use branch entities in procedures when an operation has alternate entity destinations. There are some cases where traversal is not deterministic, for example, when an operation has alternate destinations defined. These alternate destinations can be added as branch entities in a procedure. When the operation is executed at runtime, the path the procedure takes is determined by which branch entity is recognized after the operation completes. If none of the branch entities is recognized, the procedure fails.

- Adding Error Entities to a Procedure

  Error Entities are screens containing patterns that indicate an error has occurred in the procedure. You can define error entities by purposely entering bad data in a host application and capturing the resulting screen as an entity. Adding one or more error entities to a procedure is a good way to build error checking into your model.

More information

Inserting a Recordset

Inserting an Error Entity

Inserting a Branch Entity

4.6.5 Creating Descriptions for Entity Definitions

The following options allow you to include descriptions in any exported documentation. This data is included in Web Builder projects or any documentation generated using Export options. This applies to Advanced Attribute Properties, Advanced Operation Properties, Advanced Recordset Properties, and Advanced Recordset Field Properties.

Click Advanced Properties next to the **Name** box on the corresponding Attribute, Operation, Recordset, and Recordset Field tabs to open these dialog boxes.

4.7 Adding Operations to an Entity

After you've created an entity and have added patterns and attributes to that entity, you're ready to add or edit an operation.

Tip

Object names for entities, attributes, operations, recordsets, recordset fields, and, variables are not case sensitive.

To add or edit an operation:

On the Operation tab, click the **New Operation** button in the **Name** box. By default, the operation is named `Operation_1`.

To change the name, select it and type a new name in the **Name** box.

Configure your new operation in the **Destination** box, **Command list** box, and **Properties** boxes.

After configuring your operation, click **Apply** to save your changes.

To copy an operation:

You can copy the operation to create another that is similar.

Select the operation you want to copy.

Click the list next to the **New Operation** button and select **Create Copy**. A new operation with a similar name and a number appended to the end is created (for example, `ToMainMenu_2`).

Modify the new operation so that it is distinguishable from the operation on which it is based.

More information

Adding entities to a model

4.7.1 Mapping Operations

The Design Tool provides several commands that enable advanced configuration of entities in the host application model. To edit an existing operation, click the **Operation** tab and then click the **Edit** button to open the **Operation Edit** dialog box.

See Operation command summary for a complete list of available commands.

4.7.2 Reset Current Recordset

This operation can be used when multiple database records have the same entity definition but scrolling changes entity attributes and entity recordset contents. To ensure that new records are fetched, add the **Reset Recordset** command in a scrolling operation before the actual keystroke that performs the scroll.

To use Reset Current Recordset:

1. On the Operation tab, create a Down operation and rename it `DownReset`. If you already have a Down operation on the entity, you can click **Copy Operation** and rename it `DownReset`. For an IBM host, the operation looks like this.

   `CheckOperationConditions`

   `TransmitTerminalKey rcIBMpf2Key`

> **Note**
>
> For VT, the terminal key is `VTNextPage`; for HP, the terminal key is `HPPageDown`.

2. Add the ResetRecordset command as the second line of the DownReset operation. The operation for an IBM recordset reset looks similar to the following:

   `CheckOperationConditions`

   `ResetRecordset "<recordsetname>"`

   `TransmitTerminalKey rcIBMpf2Key`

More information

Adding Recordsets to an Entity

4.8 Tables and Procedures

4.8.1 Overview

Using tables and procedures in your host application model enables you to create a database abstraction of the host data. Client applications can then query this data using a Verastream connector to interact with the model.

A table is a structure that contains columns that are used as input and output parameters for procedures. Typically, table columns are named to match their corresponding model attributes and fields.

A procedure defines how Host Integrator locates, retrieves, updates, inserts, and/or deletes data when it fulfills a request submitted by a client application using a Host Integrator API.

Tables and procedures are interlocked functionally. Tables are a way of "organizing" host data into a database-like view of the data, and procedures manipulate that data. The only way to access the abstracted table data is through a procedure.

Client applications interact with tables and procedures using either a subset of the industry standard Structure Query Language (SQL), or by interacting directly with the procedures. You can access the data in your host applications via SQL queries even if your host applications are not designed to respond to SQL queries.

SQL Overview

With the Table/Procedure feature, you can abstract your host application so that client applications can perform queries using a subset of the industry standard Structure Query Language (SQL). This makes it possible for you to access the data in your host applications via SQL queries, even if your host application is not designed to respond to SQL queries.

In a query statement, client applications can select, update, insert, and delete data in the host application by specifying a context, a set of input parameters, and a set of desired output parameters in the form of an SQL statement. Using this statement Host Integrator determines the proper query to run and returns the desired results.

Use the table option to set up a table definition consisting of a set of columns. Use procedures to tell Host Integrator how to navigate to the host application screens where the data resides and pass any commands to the host application necessary to select, update, insert, or delete the host data. Once abstracted, the host application model responds to SQL queries from client applications in exactly the same way that a true SQL database does.

Note

Procedure execution does not require SQL--see Executing Procedures Using Connector APIs.

How Host Integrator Fulfills SQL Queries

When the Host Integrator receives an SQL query from a client application, it determines which procedure or set of procedures it must use to satisfy the query, and then executes those procedures.

You can use any VHI procedure type (SELECT, UPDATE, DELETE, or INSERT) to modify host data, but only a SELECT procedure type can return data.

For SELECT statements, Host Integrator will use the necessary procedures to return set of data that exactly matches the WHERE clause in the query. Any data that does not exactly match the WHERE clause is thrown out during a process known as post-fetch filtering. This filtering is not used for LIKE expressions, thus all data found by the procedure concerning a LIKE expression is returned. This implementation of LIKE diverges from the SQL-92 standard.

SQL Syntax

Host Integrator supports a subset of the SQL 92 standard for SELECT, UPDATE, INSERT, and DELETE statements. This section describes the syntax convention that Host Integrator supports.

Case Sensitivity

The SQL-92 language standard requires that the names of objects be compared without regard to letter case. Comparisons between column values, however, are case sensitive by default.

If you do not see the results you expect because of case sensitivity, you can add COLLATE CASE_INSENSITIVE to explicitly specify a case-insensitive comparison of text column values. In this example, the state value ('ri') will be compared without case:

```
`SELECT Name, ContractDate, AcctNumber FROM Accounts WHERE MiddleInitial =
'c' AND
State = 'ri' COLLATE CASE_INSENSITIVE AND LastName = 'smith'
```

Note

SQL keywords such as JOIN or ORDERBY are recognized by Host Integrator, but not supported. You can extend Host Integrator's native SQL support using an event handler. For example, you could do the following in an event handler:

Pass in an SQL string.

Remove the portions not supported by Verastream.

Pass the remaining SQL to the model for processing.

Modify the results based on the custom extensions from the original client string.

Table and Column Names

Table and column names in Host Integrator are identified using the following convention:

`TableName=Identifier`

`ColumnName=Identifier`

`Identifier=RegularIdentifier | DelimitedIdentifier`

A regular identifier is a string of not more than 128 characters; the first character must be a letter (upper or lower case), while the rest can be any combination of upper or lower case letters, digits, and the underscore character. No SQL reserved words can be used.

A delimited identifier is any string of not more than 128 characters enclosed in double quotes. The double quote character is represented by two immediately adjacent double quotes.

Table and column names are not case sensitive.

Literals

`Literal={CharacterString | Number }`

Character strings are written as a sequence of characters enclosed in single quotes. A single quote character is represented by two immediately adjacent single quotes. Any comparisons between literals and columns must be between the same type: strings can only be compared to strings and numbers can only be compared to numbers.

Expressions

`Expression = { Expression + Expression | Expression - Expression | Expression * Expression | Expression / Expression | - Expression | ( Expression ) | Literal | ColumnName }`

Conditions

`Condition = { Condition OR Condition | Condition AND Condition | NOT Condition | (Condition) | Comparison }`

`Comparison = Expression { = | <> | < | <= | > | >= | LIKE } Expression`

`SimpleCondition = { SimpleCondition OR SimpleCondition | SimpleCondition AND SimpleCondition | (SimpleCondition) | SimpleComparison)`

`SimpleComparison = ColumnName { = | LIKE } Literal`

A distinction is made between Conditions and SimpleConditions because only SimpleConditions can be used as inputs to a procedure. SimpleConditions can be used in any WHERE clause, but Conditions may be used only in a SELECT statement's WHERE clause because the results can be filtered. Both Conditions and SimpleConditions can refer only to columns from one table. Joins and subqueries (SELECT statements inside another SQL statement) are not supported.

Note

In the examples for the SELECT, UPDATE, INSERT, and DELETE statements assume that the SQL statements used with a model can resolve to a procedure in the model.

SELECT Statement Syntax

Use the following syntax for SELECT statements (Arguments between ([ ]) are optional and those between ({ }) are required:

```
SELECT [DISTINCT] {column-list} FROM {table} [WHERE {condition}] [ORDER BY {column-list}]
```

Arguments

- [DISTINCT] -- Specifies that all rows returned be unique. If there are two identical rows, one is removed from the output.
- {column-list} -- Any valid table column name(s).
- {table} -- The name of the table.
- {condition} -- Any condition or simple condition.

SELECT statements return only those rows exactly matching the WHERE clause. The results are sorted in the order specified in the ORDER BY clause. For procedure resolution, the table name is taken from the FROM clause, the filter parameters are taken from the WHERE clause, and the outputs are taken from the SELECT and WHERE clauses.

Examples

```
SELECT * FROM Patients WHERE AdmitNum = 20000
```

```
SELECT AdmitNum, SSN FROM Patients WHERE LastName LIKE 'W'
```

```
SELECT DISTINCT AdmitNum, SSN FROM Patients WHERE LastName LIKE 'W'
```

```
SELECT * FROM Patients WHERE LastName LIKE 'W' ORDER BY AdmitNum
```

```
SELECT * FROM Patients WHERE LastName LIKE 'W' AND FirstName = 'JOHN' ORDER BY  AdmitNum
```

UPDATE Statement Syntax

Use the following syntax for UPDATE statements:

```
UPDATE {table} SET {{column} = {value} [, ...]} [WHERE {simple-condition}]
```

Arguments

- {table} -- The name of the table.
- {column} -- Any valid table column.
- {value} -- Valid values are characters and numbers.
- {Simple-condition} -- Simple conditions can contain characters and numbers and comparisons (=, <>, <, <=, >, and =).

UPDATE statements update all records matching the WHERE clause with the values in the SET clause. For procedure resolution, the table name is taken from the UPDATE clause, the filter parameters are taken from the WHERE clause, and the data parameters are taken from the SET clause.

Arguments between ([ ]) are optional and those between ({ }) are required.

Example

```
UPDATE Patients SET FirstName = 'Colin', LastName = 'Moulding', AdmitYear = 1999
WHERE AdmitNum = 56564
```

INSERT Statement Syntax

Use the following syntax for INSERT statements:

```
INSERT INTO {table} [({column-list})] VALUES {value-list}
```

Arguments

- {table} -- The name of the table.
- {Column-list} -- Any valid table column name(s). If no column list is defined, the order of the columns is taken from the table definition.
- {values-list} -- Valid values are characters and numbers.

INSERT statements add a record to the specified table. For procedure resolution, the table name is taken from the INSERT INTO clause. Data parameters are taken from the VALUES clause.

Arguments between ([ ]) are optional and those between ({ }) are required.

Examples

```
INSERT INTO Patients (AdmitNum, FirstName, LastName, Room) VALUES (31415,    'Doe', 'John', '123')
```

```
INSERT INTO Patients (AdmitNum, FirstName, LastName, Room) VALUES (31415, 'Doe',    'John', '123'),
(31416, 'Doe', 'Jane', '131')
```

DELETE Statement Syntax

Use the following syntax for DELETE statements:

```
DELETE FROM {table} [WHERE {simple-condition}]
```

Arguments

- {table} -- The name of the table.
- {Simple-condition} -- Simple conditions can contain characters, numbers, and comparisons (=, <>, <, <=, >, and >=).

The delete statement deletes a record from the specified table. For procedure resolution, the table name is taken from the DELETE FROM clause and the filters are taken from the WHERE clause.

Arguments between ([ ]) are optional and those between ({ }) are required.

Example

```
DELETE FROM Patients WHERE AdmitNum = 31415
```

SQL Syntax Restrictions

Host Integrator supports the SQL-92 arguments and features, with the following exceptions:

SELECT statements containing GROUP BY or HAVING clauses.

DEFAULT and NULL may not be used for column values in UPDATE or INSERT statements.

INSERT statements that specify inserting DEFAULT VALUES.

Nested queries are not supported.

Parameter references in expressions are not supported.

The AVG, BIT_LENGTH, CASE, CAST, CHAR_LENGTH, CHARACTER_LENGTH, COALESCE, CONVERT, COUNT, CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP, CURRENT_USER, EXTRACT, LOWER, MAX, MIN, NULLIF, OCTET_LENGTH, POSITION, SESSION_USER, SUBSTRING, SUM, SYSTEM_USER, TRANSLATE, TRIM, and UPPER functions are not supported in expressions.

JOIN operations are not supported.

The SET clause can only accept literal values.

For UPDATE and DELETE statements, all comparisons in the WHERE clause must be between columns and literals.

For SELECT statements, you can use any valid expression in the WHERE clause. However, only comparisons between columns and literals will be used in a procedure. The other expression will be used to filter the output of the procedure(s). Any data that does not exactly match the WHERE clause is thrown out during a process known as post-fetch filtering.

Host Integrator supports a usage of the LIKE operator that turns off post-fetch filtering. In this case, LIKE is equivalent to =, but post-fetch filtering is not performed on the column. if you expect to get everything the host sends, use LIKE. If you want to filter the results (including filtering for case sensitivity), use =. For example:

```
SELECT Name, ContractDate, AcctNumber FROM Accounts WHERE MiddleInitial =
'c'    AND State LIKE RI AND LastName = 'smith'
```

SQL predicate (logical test) restrictions

Only TRUE predicates are allowed. The forms IS NOT * TRUE and IS FALSE are not supported.

The BETWEEN and NOT BETWEEN predicates are not supported.

The IN and NOT IN predicates are not supported.

The LIKE operator is used to provide an input to a procedure without post-fetch filtering. The NOT LIKE and LIKE ... ESCAPE predicates are not supported.

Predicates involving quantifiers (ALL, SOME, ANY, EXISTS, UNIQUE) are not supported.

Predicates involving MATCH are not supported.

Tables Overview

Using Host Integrator tables enables you to create a database abstraction of your host application so that client applications can query it using a subset of the industry standard Structure Query Language (SQL). Table columns are usually associated with attributes and fields in the Host Integrator model. This makes it possible for you to access the data in your host applications via SQL queries even if your host application is not designed to respond to SQL queries. Host Integrator tables themselves don't contain data; rather they provide a database-like view of the underlying host data.

In a query statement, the client application specifies a table, a set of input parameters, and a set of desired output parameters in the form of an SQL statement. From this statement Host Integrator determines the appropriate procedure or procedures to run in order to return the desired results.

Host Integrator accomplishes this through the use of tables and procedures. Using the Tables dialog box, you can create one or more tables for a host application. Each table can have one or more associated procedures. The procedures are comprised of entity navigation paths and parameter mappings that tell Host Integrator how to read, write, and modify host data represented by the table.

Use the Table Wizard or the Tables dialog box to create a database abstraction of the Host Integrator model.

When you export the documentation for your host application model, a list of all table and procedure names is generated. Use this list to access the tables and procedures from one of the Host Integrator APIs

More information

Creating tables

Procedures Overview

Procedures tell Host Integrator how to fulfill the queries it receives from client applications. The procedures you create for your table determine what host data can be read, inserted, updated, or deleted. Each procedure has a unique signature that describes what it does. The signature includes a procedure type (SELECT, UPDATE, INSERT, and DELETE) and a set of parameters. Host Integrator uses these signatures to translate SQL statements into a set of procedures.

You can use any VHI procedure type (SELECT, UPDATE, DELETE, or INSERT) to modify host data, but only a SELECT procedure type can return data.

Procedures use one or two of the three types of parameters:

Filter parameters— Specify which records will be acted upon

Data parameters— Specify new values for the records

Output parameters— Specify what values to return

The key component of a procedure's definition is the parameter mapping. Each parameter in a procedure corresponds to a column in the table and is mapped to an attribute, a recordset field, or another parameter. Each procedure has a predefined traversal path through the host application; during the traversal operations, data is exchanged between parameters and attributes and recordset fields. The following chart shows which parameters are used in which procedures:

Procedures	Filter Parameters	Data Parameters	Output Parameters
SELECT	X		X
UPDATE	X	X	
INSERT		X	

Procedures	Filter Parameters	Data Parameters	Output Parameters
DELETE	X		

Procedures should be as complete as possible: if you do not provide a procedure for a particular operation, it is not be possible for a client application to access or modify that table data. Procedures should also contain robust error handling to recover from unexpected or incomplete queries. Using the Procedure Editor, you can include error entities that define errors returned from a procedure.

Use the Procedure Wizard to quickly create a basic procedure. For more complicated procedures, create the procedure using the Tables dialog box and the Procedure Editor.

After adding procedures to your model, you can use Web Builder to quickly and easily generate a web application or a component interface, such as a web service or JavaBeans, based on the procedures of a host application model.

Note

When creating procedures to be used for generating a web application with Web Builder, you must have unique procedure names throughout the model. Do not create a procedure with the same name for two different tables.

More information

- Creating procedures
- Parameter mapping
- Creating tables

4.8.2 Creating Tables

Using Host Integrator tables enables you to create a database abstraction of your host application so that client applications can query it using a subset of the industry standard Structure Query Language (SQL). Table columns are usually associated with attributes and fields in the Host Integrator model. This makes it possible for you to access the data in your host applications via SQL queries even if your host application is not designed to respond to SQL queries. Host Integrator tables themselves don't contain data; rather they provide a database-like view of the underlying host data.

In a query statement, the client application specifies a table, a set of input parameters, and a set of desired output parameters in the form of an SQL statement. From this statement Host Integrator determines the appropriate procedure or procedures to run in order to return the desired results.

Host Integrator accomplishes this through the use of tables and procedures. Using the Tables dialog box, you can create one or more tables for a host application. Each table can have one or more associated procedures. The procedures are comprised of entity navigation paths and parameter mappings that tell Host Integrator how to read, write, and modify host data represented by the table.

Use the Table Wizard or the Tables dialog box to create a database abstraction of the Host Integrator model.

When you export the documentation for your host application model, a list of all table and procedure names is generated. Use this list to access the tables and procedures from one of the Host Integrator APIs.

Creating a table using the Table dialog box

In the Design Tool, click Tables on the Model menu to open the Tables dialog box.

Then, click New in the Tables dialog box and select Table in the Create a new table or procedure dialog box that appears.

Type a name for the new table in the Name box. This is the name Host Integrator will use to identify the table in the Table Editor and that client applications use to query the host application model.

4. Type a description of the table in the Description box. To add columns to the table:

- Click the Add Column (+) button to the right of the Columns box to add one or more new columns.
- In the Name field, replace the auto-generated column name (Column1, Column2, etc) with a meaningful name, typically the name of the attribute or recordset field that this column will be mapped to in a procedure.
- Select the column's data type by clicking the down arrow in the Data Type entry area and selecting one of the options: Float, Integer, or Text.
- Click the Key box to identify this column as the key.
- If required, specify the column's minimum and maximum values in the Column properties minimum/maximum boxes.

Enter a description of the column in the Description box.

5. If you want the Host Integrator to return a partial set of data to a querying application using SQL, select the Allow SQL SELECT statements to return a subset of columns when all columns are requested box. If you clear this check box, Host Integrator returns an error to a querying application if it cannot return all columns when a wildcard `*` is specified in SQL.

Note

You can create tables using the Table Wizard, which prompts you for the necessary information.