User's Guide for the MediaGateway 2

MediaGateway 2User's Guide for the v2.0

Publish Date: May 2014

Bold Technologies Ltd
421 Windchime Place
Colorado Springs CO 80919
USA

Phone: +1-719-593-2829
Toll Free US 1-800-255-2653 (BOLD)
Fax: +1-719-599-3953
Email: sales@boldgroup.com
support@boldgroup.com

Acknowledgements

Publisher

Bold Technologies, Ltd.

Author

Bold Technologies, Ltd
Technical Writer

Table of Contents

Welcome

Bold's MediaGateway 2 is a versatile tool that offers flexibility in communication to optimize Central Station processes. The MediaGateway 2 is designed to:

- Communicate with the Bold Telephony Service (BTS) by sending commands and receiving responses. The BTS is also capable of initiating communication with the MediaGateway 2.
- Communicate with Manitou by sending and receiving commands and responses through the XML Receiver Driver.
- Use Positive Voice Detection (PVD), to determine if a human voice has answered an outbound call. The MediaGateway 2 also supports Positive Answer Machine Detection (PAMD) to detect when an answering machine has answered a call.

A Bold Introduction

This manual is intended to provide instruction and reference for core features and functions in the MediaGateway 2.

About Bold Technologies

Bold Technologies, an industry leader since 1981, has stood the test of time in Central Station and alarm automation software. Formerly known as Bradley On-Line Data Systems, Bold Technologies created the Bold Central Station Automation System. Through the years, Bold has shaped the development of contemporary alarm monitoring by introducing faster and more sophisticated software innovations. Bold promotes excellence and technological innovation through strategic partnerships and support of organizations dedicated to advancing industry standards. As a proven leader in the security industry, our strength is founded on a company-wide commitment to helping our Customer group through superb products and services. This commitment guides our decisions for future development.

About Manitou

Bold Technologies, recognizing that the future was about innovations in the Windows® world, began building the Manitou system from the ground up in 1999. The first sites went live on this new-age monitoring system in 2002. The Manitou system was completely redesigned and revolutionary, opening the alarm industry to new concepts and ideas. This endeavor completely changed the industry paradigm, and encouraged broad advancements industry-wide.

About the MediaGateway

Another Bold endeavor, the MediaGateway came on to the scene in 2006, bringing an opportunity for sites to streamline multiple communication pathways. In 2012, we revised and updated the MediaGateway by adding several new industry-leading features to the ever-expanding list of supported modules.

How to Use this Guide

This manual was meant to be viewed as a .pdf or in electronic form. Incorporated herein are overviews, key terms, and definitions as well as numerical, step-by-step instructions on how to perform different tasks in the MediaGateway 2.

Formatting Key

Formatting	Used for
<Bolded>	Bolded, Courier font letters in brackets refer to actual keys on the keyboard.
Italicized	Italicized words indicate certain titles such as *Alarm Handling* window or *Jump To* menu, as well as to provide emphasis on certain words.
Bolded	Bolded words are used to signify important data, action paths and software buttons such as **OK** or **Operations ➢ On Test Status**.

Icon Key

Icon	Used for	Description/Formatting example
💡	**Recommendation/Idea**	*Suggestions to enhance usage of software functions and features.*
🗁	**Valuable Information**	*Any type of information deemed to be valuable such as subject notes and additional information pertaining to the subject matter.*

💣	Warning/Caution	**Reminders of things NOT to do or to be careful and/or aware of.**
	Example	Provides examples to further explain topics or steps.

System Requirements

- **Manitou CS 1.6.1**
  The MediaGateway 2 requires Manitou CS 1.6.1 in order to run properly and provide its trademark flexibility in functionality.

- **SQL Express**
  SQL Express provides the means for the MediaGateway 2 to store settings and log into a database.

Getting Started

The MediaGateway 2 provides Manitou users with the ability to perform numerous tasks in conjunction with the Manitou system. Because the MediaGateway 2 can be configured with different tools to fit unique needs, it offers a winning combination of capability and functionality.

The Manitou software makes handling alarms and managing accounts a simple, efficient task. For most functions, the Operator works from the Manitou main interface. Therefore, it is very helpful to become familiar with the Manitou application.

Note: for more information on the Manitou system, refer to the appropriate manual for your particular version of the Manitou product line.

Testing

The MediaGateway 2 is installed and configured by a member of Bold's Implementation and Support team. Once the initial installation is complete, several tests may be run in order to ensure that everything was installed correctly. These tests ensure that the core functions of the system are running properly. First, however, users must check the status of the Telephony Server.

Connecting to the Bold Telephony Interface

To test if the Bold Telephony Service (also known as BTS) is running, remotely connect to the server running the Telephony service and connect over TCP/IP to the command interface of the Bold Telephony Service. Specify the port (6110) on which the service is running. For example, enter "telnet localhost 6110" from a command prompt, where localhost is the name of the server where the Bold Telephony Service is running and 6110 is the port number on which the Bold Telephony Service is listening for connections.

To test if the Bold Telephony Service is operating correctly, run the command "0,0,-HELP". This shows all available commands that the Bold Telephony Service accepts. If this test fails, users will need to troubleshoot the installation.

Executing the "0,0,-HELP" command displays a report of all available devices the Bold Telephony Service can access. In the example below, only COM ports are available. If a Dialogic board had been installed, it would have shown all ports available on that device. If there are Dialogic boards installed and the command interface does not return the port information, it likely means that the board process was started from the Dialogic

Configuration Manager (DCM). The configuration manager can be found by clicking on the "Start" menu and selecting All Programs > Intel Dialogic System Release > Configuration Manager - DCM. This only displays if the Dialogic boards and the Intel Dialogic System software have been correctly installed.

In order for Outbound OpenVoice or TwoWayPlus/InstantConnect to transfer work properly, the Telephony server must have a dialing rule set up for the default location. This is set up on the computer (independent from Manitou) in Control Panel > Phone and Modem Options. This is the only piece of TAPI that the MediaGateway uses to complete calls. Without it, the call/transfer will fail.

Delays and Handshakes

CN4 and CN8 panels have single chips that perform call progress detection. In order for the panels to distinguish if there is a busy tone or some other problem, the firmware is set to perform call progress detection for six seconds. The CN5 panels, however, have a dual chip configuration. Therefore, they can perform call progress detection and look for the handshake at the same time.

How it Works

The MediaGateway 2 communicates with the Bold Telephony Service, which acts as a server. The MediaGateway 2 sends commands to the Bold Telephony Service (BTS), and receives responses back. It is also possible for the Bold Telephony Service to initiate communication to the MediaGateway 2. Additionally, the MediaGateway 2 communicates with Manitou as a service, sending and receiving information through the XML Receiver Driver. Finally, the MediaGateway 2 uses Positive Voice Detection (PVD). PVD listens when an outbound call is answered to see if a human voice is detected. The MediaGateway 2 also supports Positive Answer Machine (PAM) detection to determine whether an answering machine has answered the call.

The heart of the MediaGateway 2 is Hierarchical Application Language (HAL). HAL is a tree-based language comprised of Manitou Functional Components (M.F.C.s). This is a simple, visual language intended for users without programming experience. HAL enables these users to employ the features of many advanced programming languages. M.F.C.s are composed of a number of properties with a single method. The method may include parameters that affect the method function, or return either a True or False response (depending on success or failure). The next appropriate action, then, is based on the success or failure of the previous method.

New Features

New Features

Social Media Monitoring - monitors Twitter and Facebook for information that could trigger an alarm. Through authorizing an account on Twitter, Social Media Monitoring can send tweets as well as monitor by search criteria within a specified area, using Geo-Fencing. While Facebook does not currently allow postings from the MediaGateway 2, the module can still monitor current postings using ranked search requests and Geo-Fencing specifications.

Geo-Fences - provides user-definable, virtual perimeters for real-world geographic areas. Users can create circular or polygon-shaped boundaries in which to perform searches on Social Media related data. The MediaGateway 2 can also use Geo-Fences to send alarms based on entry or exit from a Geo-Fence.

UniversalConnector - connects a variety of different delivery mechanisms into one location for easy setup and management. The Universal Connector allows signals to be delivered through PRI/T1/Analog, PBX Server, email, SMS Gateway, ODBC database, FTP, TCP, RSS, and simple files.

Logging into the MediaGateway 2

Logging into the MediaGateway 2 is quick and simple.

First Time Users

Start the MediaGateway 2 in either of the following two ways:

- **Background service** - allows the user to completely log off the machine, as well as to make configuration changes
- **Standalone application** - used only with initial startup to configure the system and options

Command Line Options

For command line, use the following:

- **Standalone** - for debugging and initial configuration

- **Support** - to run the MediaGateway 2 without graphics in the background

Database Errors

Since the installer does not create a database, the application may prompt you to create one. Enter a name for the new database.

Telephony Server Warning

After you have created the database, the application may present an additional error message regarding the Telephony Server. If this occurs, click "**OK**" to continue. The Bold Telephony Server will be added later in the configuration process. If the entire system has been purchased, the user has two different servers: one running the Bold Efficiency Collection, and the other running the Personal Safety Collection. Each server then functions as a backup for the other.

Logging into the MediaGateway 2

To log into the software, begin by launching the system. You can launch the MediaGateway 2 software in either of the following ways:

- Double-click the MediaGateway icon on your desktop as shown in the following screenshot:

Alternatively, navigate the following path:

- Start Menu → Programs → Bold Technologies → MediaGateway

Note: if you have trouble logging on, contact a Supervisor.

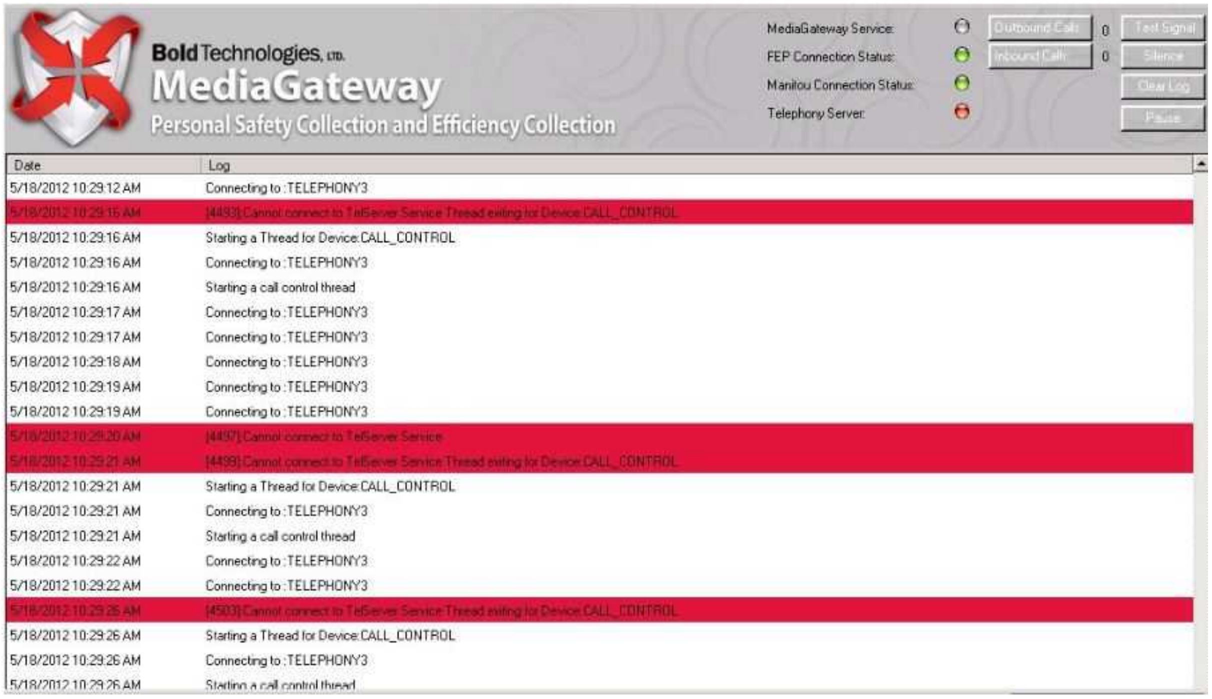
Logging On

At the login screen, enter your username and password, and then press "**Enter**". The main MediaGateway 2 screen now displays. At initial login to the MediaGateway 2, the callout screen displays which collection or collections are currently running on the system.

It is important to note that passwords for the MediaGateway 2 are configured by level. Default passwords are provided at the time of your installation.

- **Level 1** - Operator/entry level, can view log
- **Level 2** - mid-level, can view menus
- **Level 3** - highest level, can change menus

Once the MediaGateway 2 loads, it automatically launches the Activity Log. The Activity Log displays the date and time for all activities logged (both current and past). Logged activities include any actions from the MediaGateway 2 or the Telephony server.

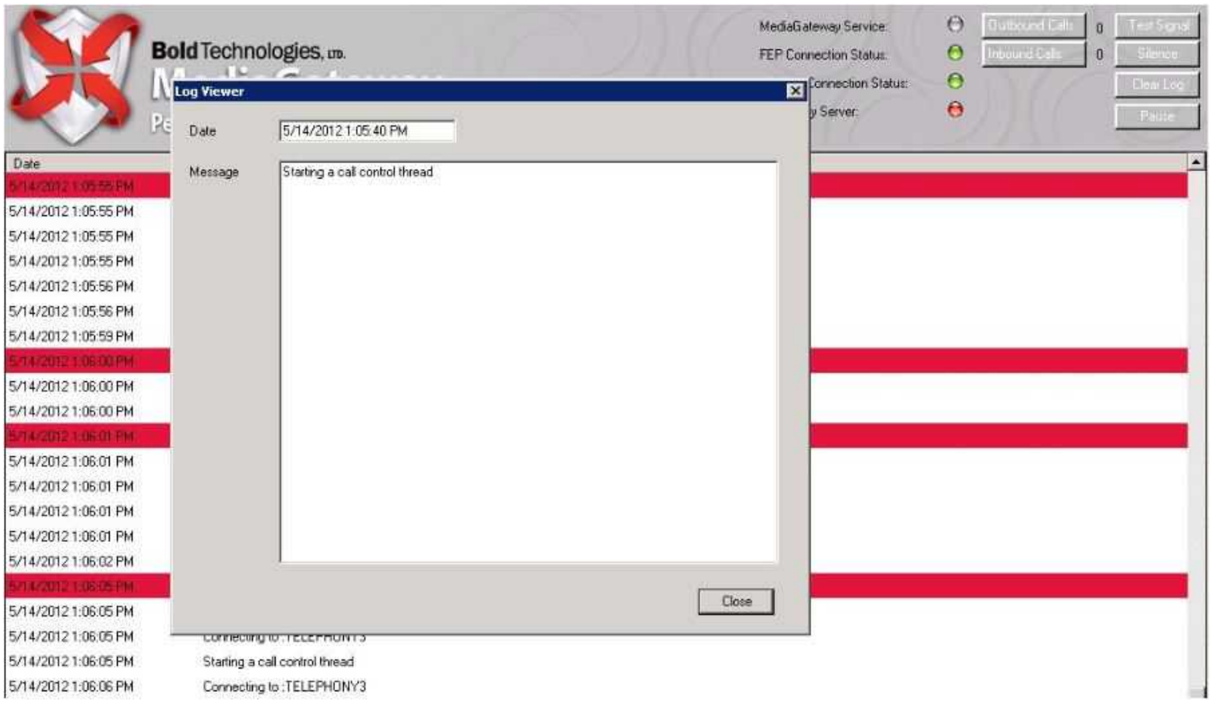


```
Date                    Log
5/18/2012 10:29:12 AM   Connecting to :TELEPHONY3
5/18/2012 10:29:16 AM   [4493] Cannot connect to TelServer Service Thread exiting for Device CALL_CONTROL
5/18/2012 10:29:16 AM   Starting a Thread for Device:CALL_CONTROL
5/18/2012 10:29:16 AM   Connecting to :TELEPHONY3
5/18/2012 10:29:16 AM   Starting a call control thread
5/18/2012 10:29:17 AM   Connecting to :TELEPHONY3
5/18/2012 10:29:17 AM   Connecting to :TELEPHONY3
5/18/2012 10:29:18 AM   Connecting to :TELEPHONY3
5/18/2012 10:29:19 AM   Connecting to :TELEPHONY3
5/18/2012 10:29:19 AM   Connecting to :TELEPHONY3
5/18/2012 10:29:20 AM   [4497] Cannot connect to TelServer Service
5/18/2012 10:29:21 AM   [4498] Cannot connect to TelServer Service Thread exiting for Device CALL_CONTROL
5/18/2012 10:29:21 AM   Starting a Thread for Device:CALL_CONTROL
5/18/2012 10:29:21 AM   Connecting to :TELEPHONY3
5/18/2012 10:29:21 AM   Starting a call control thread
5/18/2012 10:29:22 AM   Connecting to :TELEPHONY3
5/18/2012 10:29:22 AM   Connecting to :TELEPHONY3
5/18/2012 10:29:26 AM   [4503] Cannot connect to TelServer Service Thread exiting for Device CALL_CONTROL
5/18/2012 10:29:26 AM   Starting a Thread for Device:CALL_CONTROL
5/18/2012 10:29:26 AM   Connecting to :TELEPHONY3
5/18/2012 10:29:26 AM   Starting a call control thread
```

MediaGateway, Activity Log

Log Viewer

Use the Log Viewer to isolate an event in the log. To view details on a specific event, double-click the log entry.



Date: 5/14/2012 1:05:40 PM

Message: Starting a call control thread

```
Date
5/14/2012 1:05:55 PM
5/14/2012 1:05:55 PM
5/14/2012 1:05:55 PM
5/14/2012 1:05:55 PM
5/14/2012 1:05:56 PM
5/14/2012 1:05:56 PM
5/14/2012 1:05:59 PM
5/14/2012 1:06:00 PM
5/14/2012 1:06:00 PM
5/14/2012 1:06:00 PM
5/14/2012 1:06:01 PM
5/14/2012 1:06:01 PM
5/14/2012 1:06:01 PM
5/14/2012 1:06:01 PM
5/14/2012 1:06:01 PM
5/14/2012 1:06:02 PM
5/14/2012 1:06:05 PM
5/14/2012 1:06:05 PM
5/14/2012 1:06:05 PM   Connecting to :TELEPHONY3
5/14/2012 1:06:05 PM   Starting a call control thread
5/14/2012 1:06:06 PM   Connecting to :TELEPHONY3
```

Operations/Activity form, Log Viewer

Note: everything that displays in the MediaGateway 2 Activity Log is also recorded in the Manitou log.

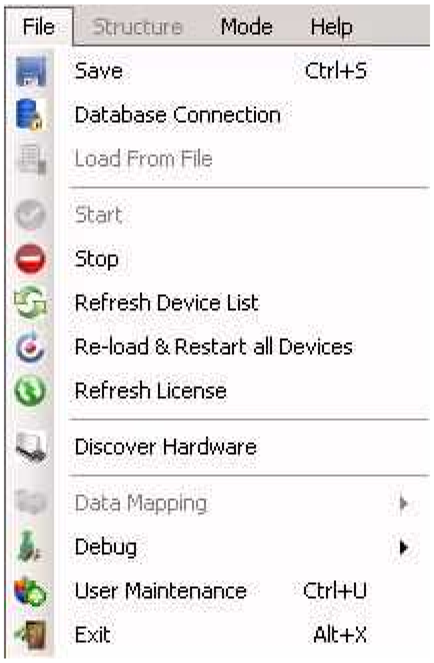
Pulldown Menus

The MediaGateway 2 pulldown menus offer easy access to options for monitoring, maintenance, connections, and modes.

Notes: items on pulldown menus may display as grayed-out depending on the specific form and actions you selected.

File Menu

The File menu includes basic commands necessary for running the MediaGateway 2.

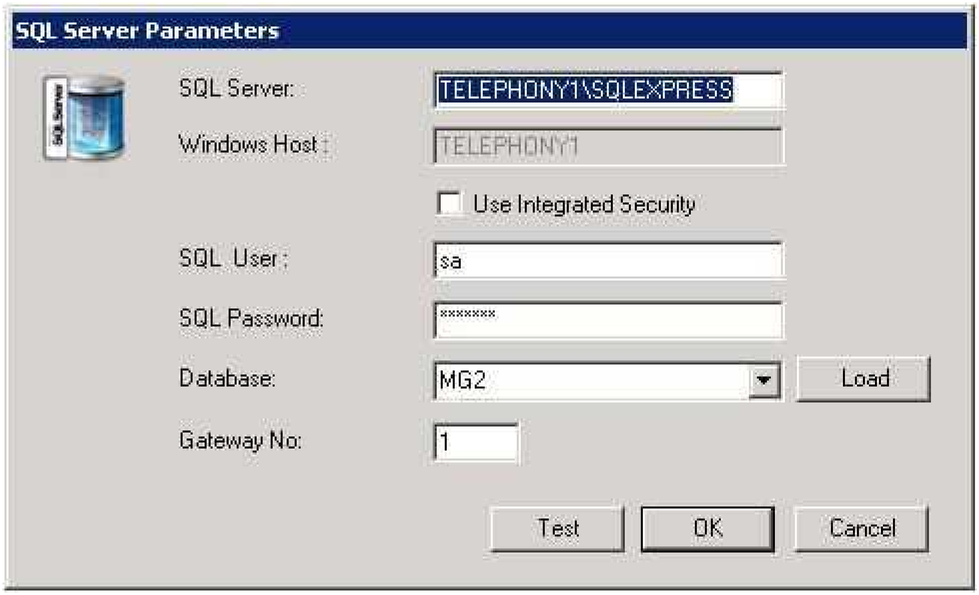


Save Menu

The "Save" menu command saves any configuration changes you made. This includes changes to Line Drivers, menu items, data maps, and options. Users must save configuration changes for them to become effective.

Database Connection

The "Database Connection" menu command allows you to modify settings for the database connection. The database stores your settings. When you load the MediaGateway 2, the application reads the database to determine the correct configuration.



SQL Server: TELEPHONY1\SQLEXPRESS
Windows Host: TELEPHONY1
☐ Use Integrated Security
SQL User: sa
SQL Password: *******
Database: MG2
Gateway No: 1

Database Connection, SQL Server Parameters

Modifying Database Connection Settings

1. Enter server information in the "SQL Server Parameters" window. The "SQL Server:" field requires both the server name and the SQL instance name in the following format: "server\instance" (as displayed in the screenshot above).
2. Enter the "Windows Host:" name. If you want, select the "Use Integrated Security" checkbox.

Note: using integrated security means using Windows authentication. Bold does not recommend using integrated security. Check with your Central Station Manager before making this selection.

3. Enter SQL User and SQL Password information.
4. Select the database you need from the "Database:" dropdown list, and specify a "Gateway No:".
5. Click "**Test**".
6. Repeat steps 1-5 to add all the servers you need for replication.

Load from File

Use the "Load from File"menu option to load the configuration from an XML file instead of from the database. Each time a change is made to the configuration, the options are saved to an XML configuration file. This way, you can go back to an earlier working configuration or send the configuration to Bold Support for analysis. The backup files can be found in the C:\ProgramData\Bold Technologies\MediaGateway directory.

Note: use the "Load from File:" menu option only after you use the "Stop" command to disable processing.

Start/Stop

The Start command starts the MediaGateway2 processing. The Stop command stops processing.

Refresh Device List

Use this menu option after adding new devices to the configuration.

Re-load & Restart all Devices

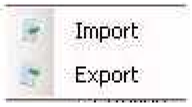
The "Re-load & Restart all Devices" menu option reloads the configuration, stops all background drivers, and then restarts the drivers. It also loads the MediaGateway 2 as a background service.

Discover Hardware

The "Discover Hardware" menu option identifies new devices it discovers on the Telephony server.

Data Mapping

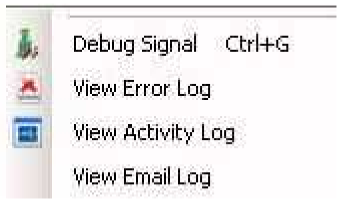
The "Data Mapping" menu option allows you to import and export data maps.



File, Data Mapping menu

Debug

The "Debug" menu options provides access to tools to assist with debugging signal input.



File, Debug menu

User Maintenance

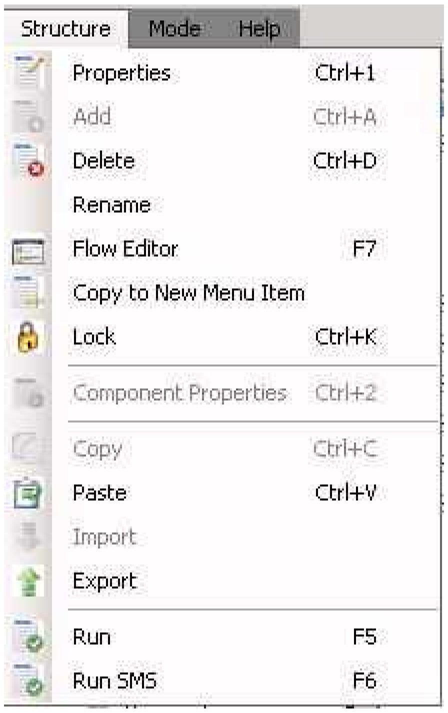
User Maintenance provides an additional security layer for users who access the system from the application layer. For example, if a person calls into the OpenVoice system, additional security can be tied to his login. The same can be done for SMS texting.

Exit

The "Exit" command exits the user interface.

Structure Menu

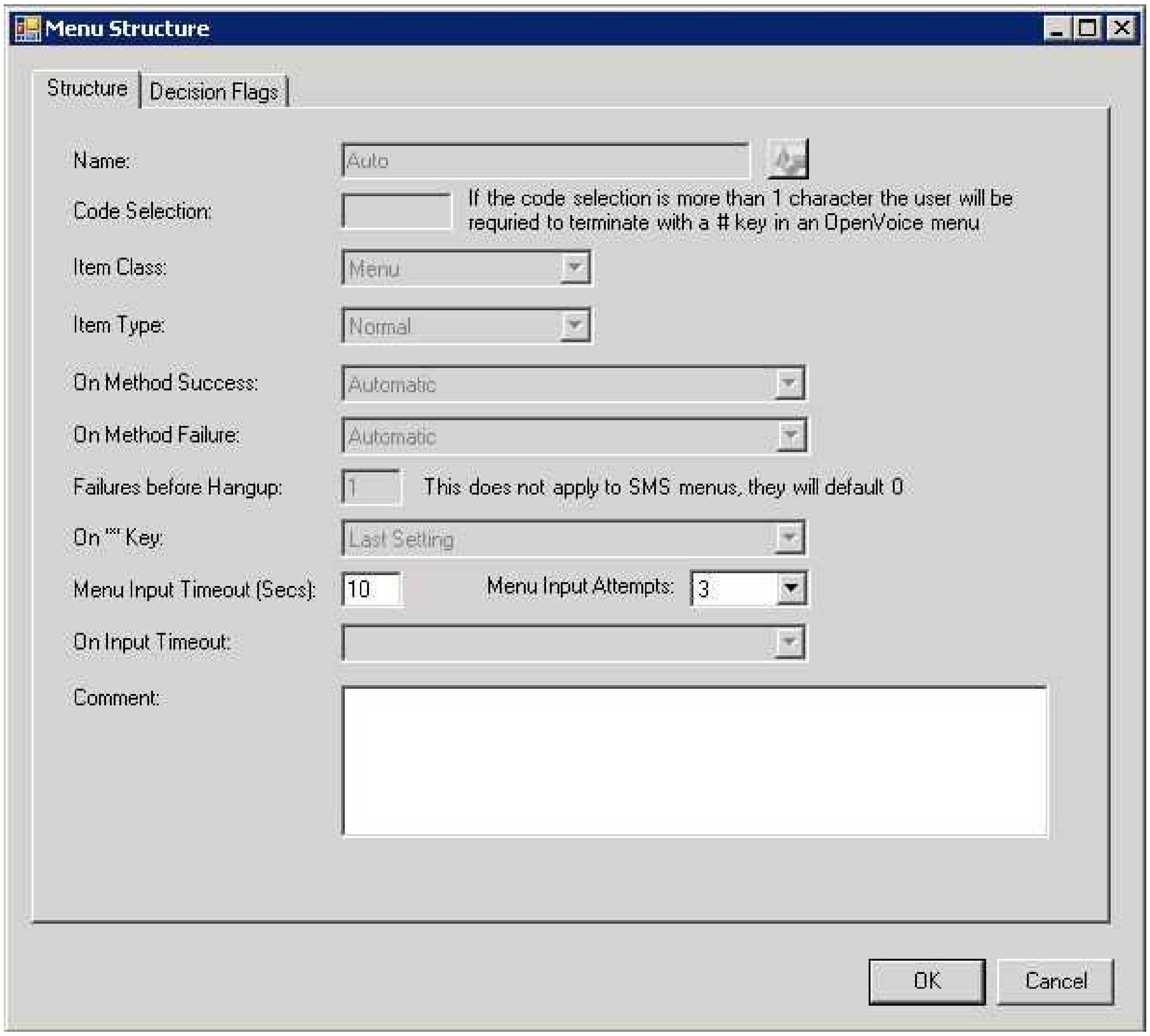
The Structure menu includes commands for adding, deleting, importing, and exporting XML files, etc., as shown in the following screenshot:



MediaGateway Structure pull-down menu

Properties

Clicking the "Properties" menu item displays the "Menu Structure" form as shown in the following screenshot:



Name: Auto

Code Selection: If the code selection is more than 1 character the user will be requried to terminate with a # key in an OpenVoice menu

Item Class: Menu

Item Type: Normal

On Method Success: Automatic

On Method Failure: Automatic

Failures before Hangup: 1 This does not apply to SMS menus, they will default 0

On "*" Key: Last Setting

Menu Input Timeout (Secs): 10 Menu Input Attempts: 3

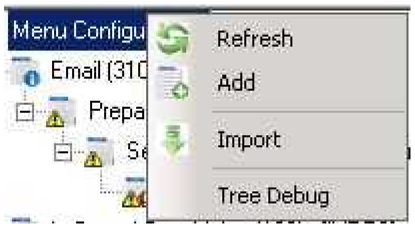
On Input Timeout:

Comment:

You can also access the "Menu Structure" form by right-clicking on a menu action. Use the "Menu Structure" form to name the menu action, apply code selection, and determine the action based on success or failure, etc.

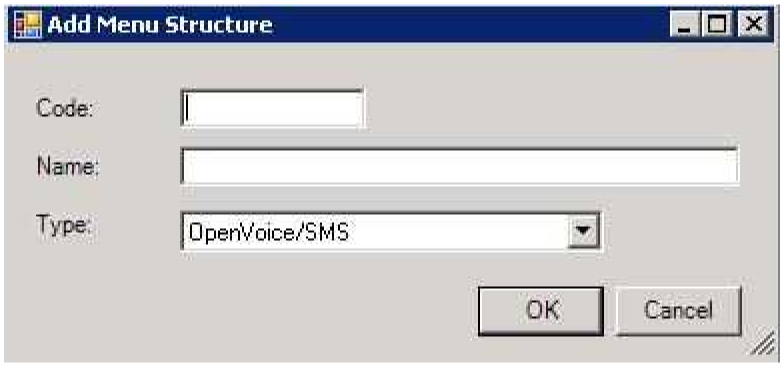
Add

Use the "Add" menu option to add a new menu structure. You can also access the "Add" command by right-clicking on "Menu Configurations" from "Menu Items" as shown in the following screenshot:



Structure Menu, Add

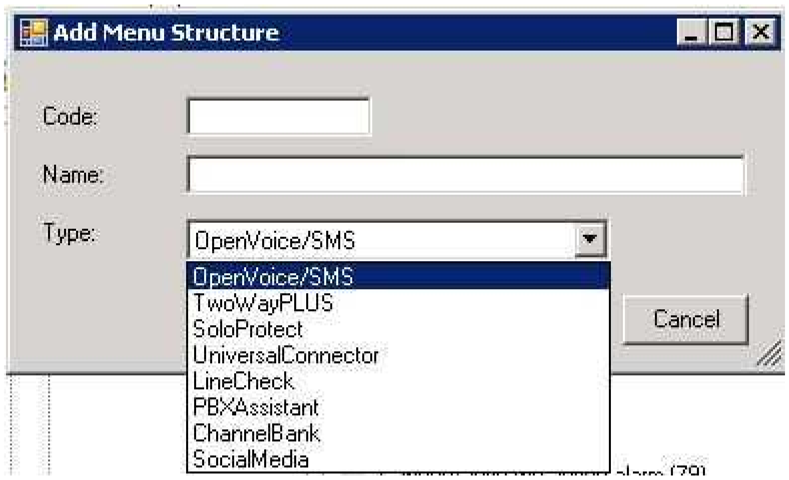
Clicking "Add" displays the "Add Menu Structure" window as shown in the following screenshot:



Add Menu Structure

Enter the following information to the appropriate "Add Menu Structure:" fields:

1. Enter the code for the menu you want to add. This code displays in the Menu field of the Line Driver.
2. Enter a name for the menu.
3. Select a menu "Type:" from the dropdown menu as shown in the following screenshot:



4. Click "**OK**", and then click "**Save**" from the File menu.

Delete

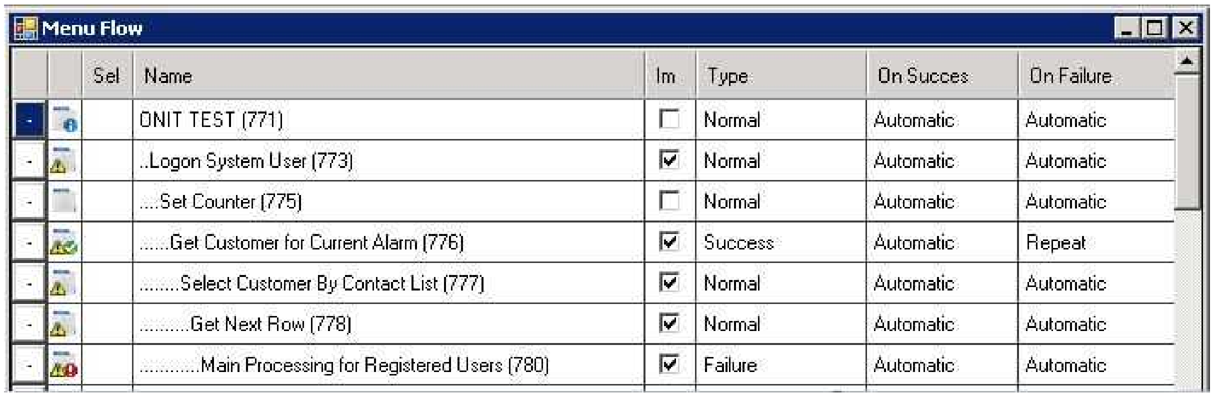
Click "Delete" to remove the selected item.

Rename

Clicking "Rename" displays the "Add Menu Structure" window. This allows you to enter a new name or to select a new menu type.

Flow Editor

Clicking the "Flow Editor" menu option displays the "Menu Flow" window as shown in the following screenshot:



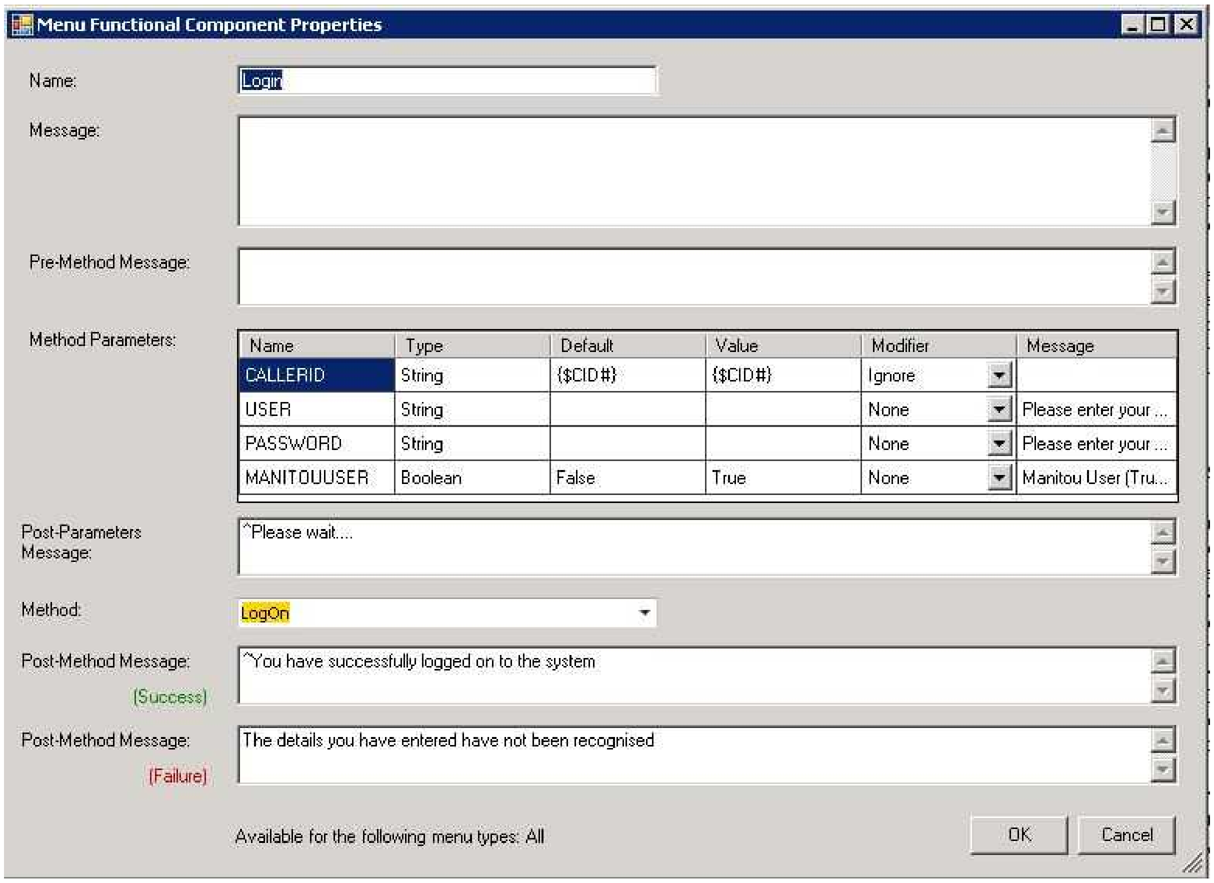
		Sel	Name	Im	Type	On Succes	On Failure
-			ONIT TEST (771)	☐	Normal	Automatic	Automatic
-			..Logon System User (773)	☑	Normal	Automatic	Automatic
-			....Set Counter (775)	☐	Normal	Automatic	Automatic
-			......Get Customer for Current Alarm (776)	☑	Success	Automatic	Repeat
-			........Select Customer By Contact List (777)	☑	Normal	Automatic	Automatic
-			..........Get Next Row (778)	☑	Normal	Automatic	Automatic
-			............Main Processing for Registered Users (780)	☑	Failure	Automatic	Automatic

The "Menu Flow" window allows you to view the contents of an entire menu in a single form. It also allows you to edit M.F.C.s while viewing how the changes you are making affect the rest of the menu. Finally, the "Menu Flow" window lets you view whether your actions result in success or failure. Successful responses from a menu action display highlighted in green, while failed responses display highlighted in red.

To access the "Menu Flow" window, select the menu item you want to edit from the "Menu Items" option in the MediaGateway 2 Navigation Tree. Then, click "Flow Editor" from the "Structure" pulldown menu.

Copy to New Menu Item

Clicking the "Copy to New Menu Item" option displays the "Menu Functional Component Properties" window as shown in the following screenshot:



Name: Login

Message:

Pre-Method Message:

Method Parameters:

Name	Type	Default	Value	Modifier	Message
CALLERID	String	{$CID#}	{$CID#}	Ignore	
USER	String			None	Please enter your ...
PASSWORD	String			None	Please enter your ...
MANITOUUSER	Boolean	False	True	None	Manitou User (Tru...

Post-Parameters Message: ^Please wait....

Method: LogOn

Post-Method Message: (Success) ^You have successfully logged on to the system

Post-Method Message: (Failure) The details you have entered have not been recognised

Available for the following menu types: All

Use the "Copy to New Menu Item" option to copy existing menu processes to a newly-created menu item.

To access the "Menu Functional Component Properties" window, select the menu from which you want to copy properties from the "Menu Items" option in the MediaGateway 2 Navigation Tree. Then, click "Copy to New Menu Item" from the "Structure" pulldown menu.

Lock

Clicking the "Lock" option also displays the "Menu Functional Component Properties" window. Use this option to lock a menu from editing, and to require entry of the menu administration password to unlock the menu.

To access the "Menu Functional Component Properties" window, select the menu you want to lock from the "Menu Items" option in the MediaGateway 2 Navigation Tree. Then, select "Lock" from the "Structure" pulldown menu.

Component Properties

Clicking the "Component Properties" option also displays the "Menu Functional Component Properties" window. Use this option to edit method parameters (for e.g., requiring the entry of a User ID and Password), and the text you want to display to the user on the success or failure of an action.

To access the "Menu Functional Component Properties" window, select the menu you want to lock from the "Menu Items" option in the MediaGateway 2 Navigation Tree. Then, click "Lock" from the "Structure" pulldown menu.

Copy and Paste

Clicking the "Copy" option copies the selected menu item and all its associated actions to the clipboard. Clicking "Paste" transfers the copied information to the selected destination.

Import

Use the "Import" command to import a menu template or a previously saved menu.

Export

Use the "Export" command to export a menu item to an XML file. We advise using this option to export and backup menus to protect against data loss.

Run

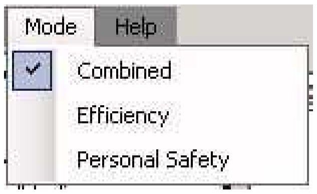
Use the "Run" option to test the menu. Clicking "Run" displays the "Menu Test" window. Once you have entered the necessary information, click "**OK**" to test the menu.

Run SMS

Use "Run SMS" to run the menu as an SMS test.

Mode Menu

The mode in which you run the MediaGateway 2 depends on the type of collection you are using.
Your three options are: "Efficiency", "Personal Safety", and "Combined" (i.e., a combination of both Efficiency and Personal Safety) as displayed in the following screenshot:

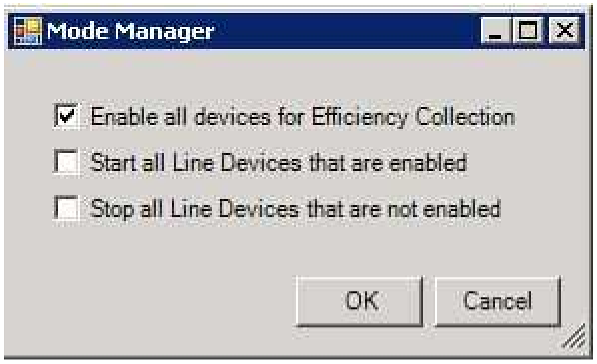


Mode pull-down menu

The Efficiency Collection of the MediaGateway 2 software does not include two-way audio functionality. The Personal Safety Collection and the Combined versions of the software include two-way audio. Refer to the "MediaGateway 2 Components" document for more information about modes.

1. To switch modes, click the option you want from the "Mode" pulldown menu.

Result: the "Mode Manager" dialog displays as shown in the following screenshot:


Mode Manager

☑ Enable all devices for Efficiency Collection

☐ Start all Line Devices that are enabled

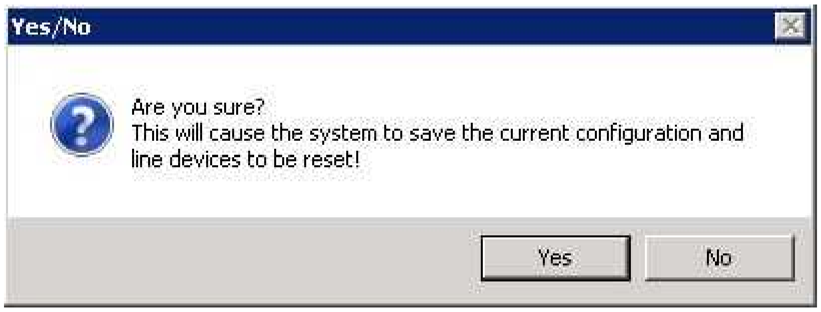
☐ Stop all Line Devices that are not enabled

Mode Manager

Note: the option to switch between collections is only available if both collections have been installed.

2. Select the option you want, and click "**OK**".

Result: the "Yes/No" confirmation dialog displays as shown in the following screenshot:


Yes/No

Are you sure?
This will cause the system to save the current configuration and line devices to be reset!

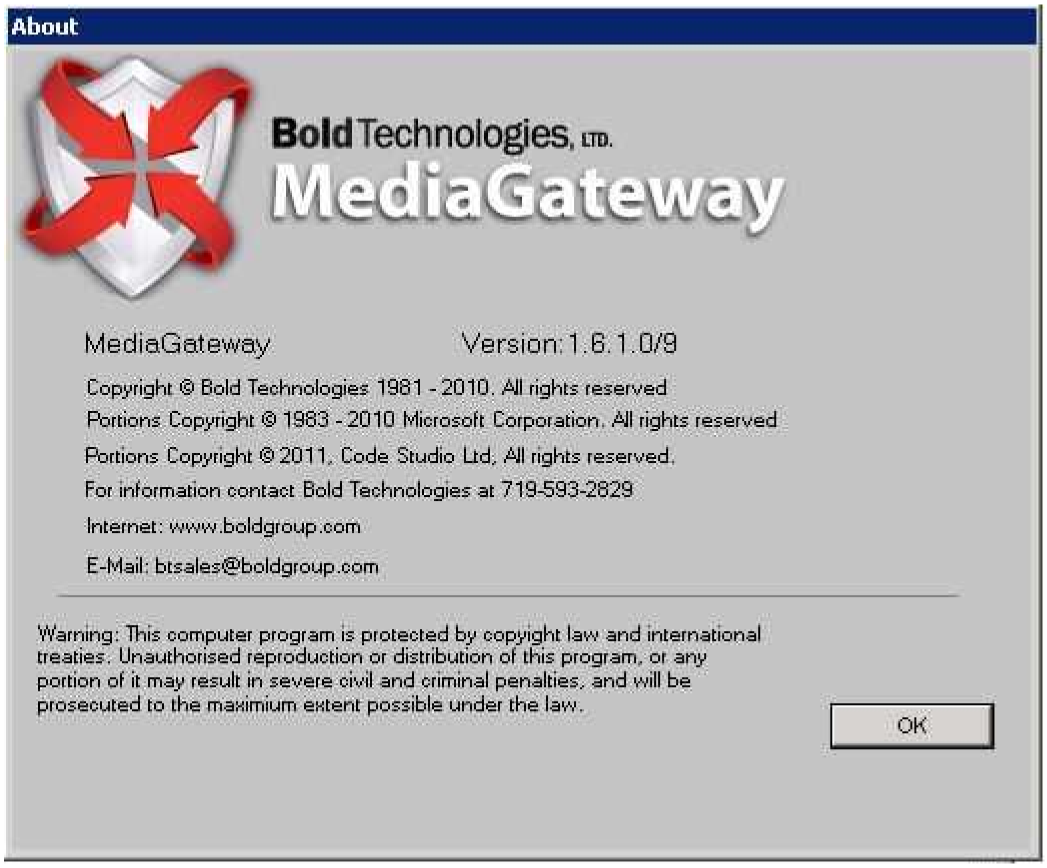
Mode change confirmation

3. Click "**Yes**" if you want to continue with the mode change. Understand that continuing with the mode change will result in the resetting of all enabled line devices.

Note: changing modes allows you to run each program side in a different mode. Therefore, if a failure occurs, you can revert to Combined mode and run both collections on one system.

Help Menu

The Help Menu displays information about the current MediaGateway 2 system you have installed as shown in the following screenshot:



MediaGateway Version: 1.6.1.0/9

Copyright © Bold Technologies 1981 - 2010. All rights reserved

Portions Copyright © 1983 - 2010 Microsoft Corporation. All rights reserved

Portions Copyright © 2011, Code Studio Ltd, All rights reserved.

For information contact Bold Technologies at 719-593-2829

Internet: www.boldgroup.com

E-Mail: btsales@boldgroup.com

Warning: This computer program is protected by copyright law and international treaties. Unauthorised reproduction or distribution of this program, or any portion of it may result in severe civil and criminal penalties, and will be prosecuted to the maximium extent possible under the law.

MediaGateway Functionality

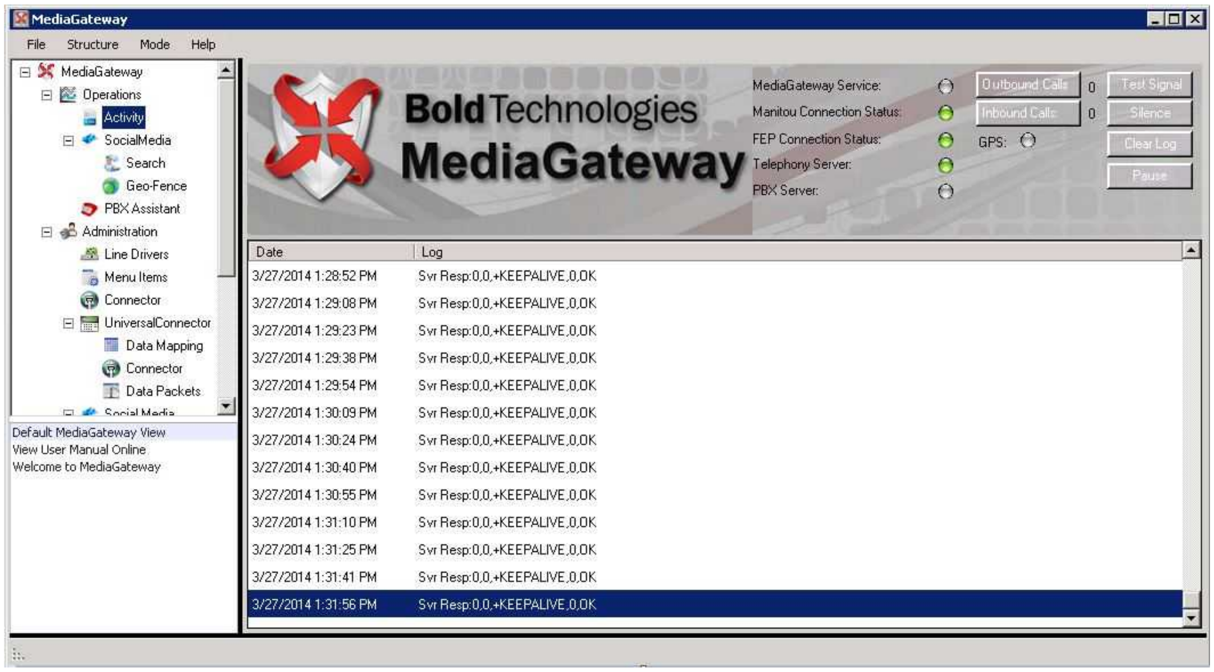
The functionality for the MediaGateway 2 can be broken down into two primary functions:

- Operations - contains forms necessary to execute tasks from monitoring and investigating activity to performing Social Media and Geo-Fence searches.
- Administration - contains forms pertaining to the setup and configuration of your settings options.

Operations

Activity Log

Operations is the primary view of the MediaGateway 2. When you launch the software, the Activity Log displays as shown in the following screenshot:



```
Date                  Log
3/27/2014 1:28:52 PM  Svr Resp:0,0,+KEEPALIVE,0,OK
3/27/2014 1:29:08 PM  Svr Resp:0,0,+KEEPALIVE,0,OK
3/27/2014 1:29:23 PM  Svr Resp:0,0,+KEEPALIVE,0,OK
3/27/2014 1:29:38 PM  Svr Resp:0,0,+KEEPALIVE,0,OK
3/27/2014 1:29:54 PM  Svr Resp:0,0,+KEEPALIVE,0,OK
3/27/2014 1:30:09 PM  Svr Resp:0,0,+KEEPALIVE,0,OK
3/27/2014 1:30:24 PM  Svr Resp:0,0,+KEEPALIVE,0,OK
3/27/2014 1:30:40 PM  Svr Resp:0,0,+KEEPALIVE,0,OK
3/27/2014 1:30:55 PM  Svr Resp:0,0,+KEEPALIVE,0,OK
3/27/2014 1:31:10 PM  Svr Resp:0,0,+KEEPALIVE,0,OK
3/27/2014 1:31:25 PM  Svr Resp:0,0,+KEEPALIVE,0,OK
3/27/2014 1:31:41 PM  Svr Resp:0,0,+KEEPALIVE,0,OK
3/27/2014 1:31:56 PM  Svr Resp:0,0,+KEEPALIVE,0,OK
```

The Activity Log shows the status of systems related to the MediaGateway 2. These system status indicators display as LED lights, and can be found in the upper-right corner of your screen. The following indicators are located on the Activity Log:

MediaGateway Service - the indicator displays green if the MediaGateway 2 is loaded as a service running in the background. The indicator displays gray if the MediaGateway 2 was loaded in Standalone mode.

FEP Connection Status - the indicator displays green if the FEP was set up with an XML Driver and is connected to the MediaGateway 2. Otherwise, the indicator displays red.
Manitou Connection Status - the indicator displays green if the MediaGateway 2 can communicate with the Manitou Broker. Otherwise, the indicator displays red.
Telephony Server - the indicator displays green if communication with the Telephony Server has been established. Otherwise, the indicator displays red.
PBX Server - the indicator displays green if communication with the PBX Server is currently occurring. Otherwise, it displays gray.
GPS - the indicator displays green if communication with the Location server is currently occurring. Otherwise, it displays gray.

The following buttons are located on the Activity Log:

Outbound Calls/Inbound Calls - the numbers to the right of the button display the number of calls the MediaGateway 2 is currently handling. Click either button to view the Call Queue.
Test Signal - sends a signal through the XML Driver into the FEP to confirm that communication has been established and is working properly. Click the button to send a test signal.
Silence - when the MediaGateway 2 loses connection with the FEP, the system emits a beeping sound. Click this button to mute the beeping.
Clear Log - clears the the Activity Log.
Pause - pauses the Log to keep it from scrolling. The button displays red when the Log is paused. Any new Log messages received while the Activity Log is paused, will be lost. The Pause feature has a revert time of two minutes (i.e., the Activity Log automatically reverts to an active state when left in a paused state for two minutes).

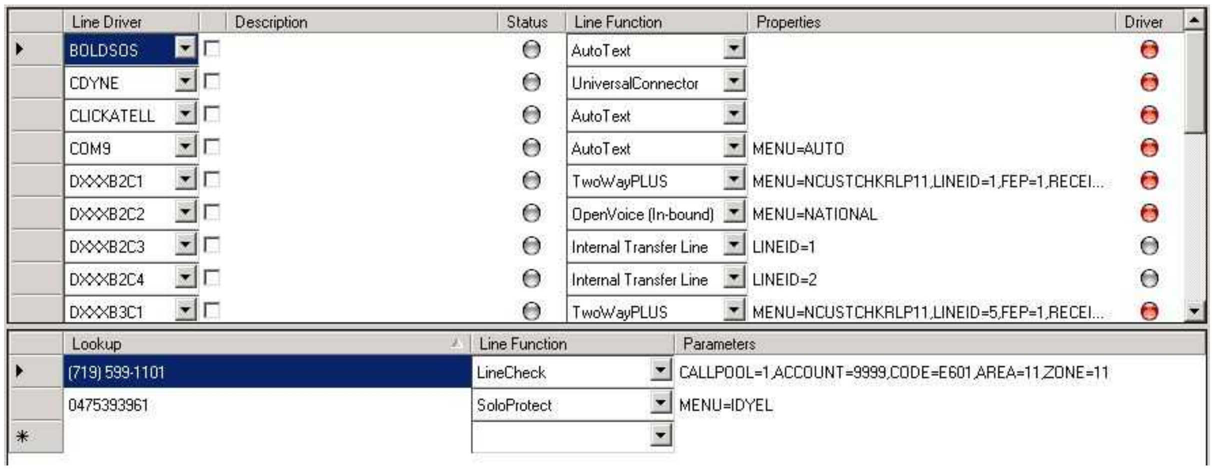
Administration

Administration is the second main view in the MediaGateway 2. From here, users can set up Line Drivers, create menus, and perform overall configuration setup. Please refer to the subtopics addressed in this manual for more specific information regarding Administration functionality.

Line Drivers

Line Drivers are used to define how the system performs according to set criteria. For example, you can assign a menu to a specific caller ID or DNIS number. Then, when the system recognizes a specific number, the menu you assigned overrides any previously defined menu.

The following screenshot provides a general view of the Line Driver window:



Line Driver		Description	Status	Line Function	Properties	Driver
BOLDSOS	☐		⚪	AutoText		🔴
CDYNE	☐		⚪	UniversalConnector		🔴
CLICKATELL	☐		⚪	AutoText		🔴
COM9	☐		⚪	AutoText	MENU=AUTO	🔴
DXXXB2C1	☐		⚪	TwoWayPLUS	MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	🔴
DXXXB2C2	☐		⚪	OpenVoice (In-bound)	MENU=NATIONAL	🔴
DXXXB2C3	☐		⚪	Internal Transfer Line	LINEID=1	⚪
DXXXB2C4	☐		⚪	Internal Transfer Line	LINEID=2	⚪
DXXXB3C1	☐		⚪	TwoWayPLUS	MENU=NCUSTCHKRLP11,LINEID=5,FEP=1,RECEI...	🔴

Lookup	Line Function	Parameters
(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
0475393961	SoloProtect	MENU=IDYEL
*		

A Line Driver can be assigned to a port on the Dialogic card, or it can be assigned to a COM port on the same machine as the Telephony server.

Note: only validly licensed items are available for selection from the Line Function menu.

Setting up Line Drivers

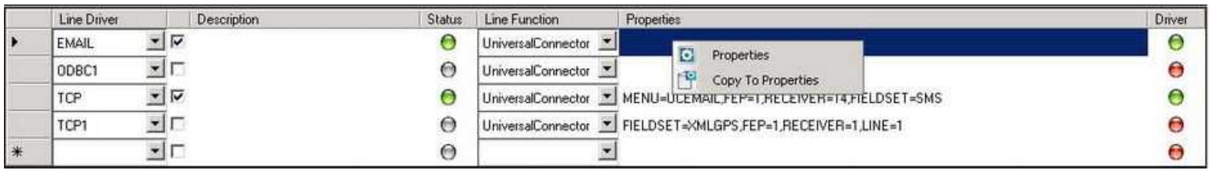
Before you set up Line Drivers, you must identify the type of driver you want and configure its properties.

Perform the following steps to configure Line Driver properties:

1. Select the Line Driver you want to associate with a Line Function from the dropdown menu in the Line Driver column.
2. Enter a description in the "Description:" field.

Note: the Status LEDs represent the color-coded status of the device. These colors mean different things for different types of Line Drivers. Typically, a green status indicates that a high signal level is present, while yellow indicates a low signal level. A red status indicates that no signal level is present. However, regarding telephone lines, a green status means the line is active (i.e., currently engaged for communications), a yellow status means the line is ready to answer, and a red status indicates an error. Additionally, a gray status indicates an inactive status (for e.g., an outbound line not currently engaged).

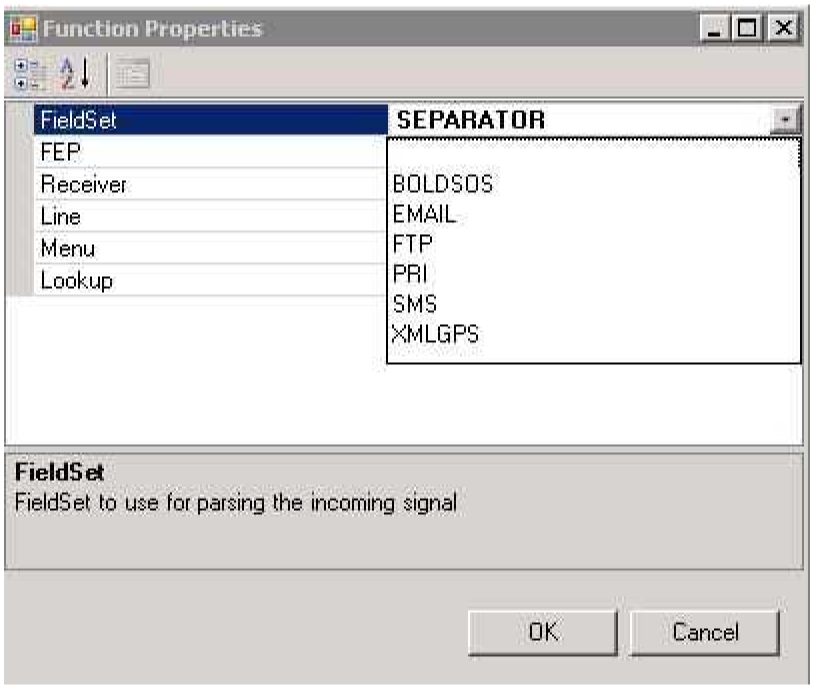
3. Select the Line Function from the dropdown menu. In the Line Driver properties example below, notice that EMAIL has been selected with a Line Function of UniversalConnector. Line Functions are licensed functions that can be associated with a device. Users can attach a function to each line or device in the system.
4. Right-click in the Properties field for the row containing your Line Driver. You can designate properties directly, or copy them from another, existing Line Driver.



	Line Driver		Description	Status	Line Function	Properties	Driver
▸	EMAIL	✓		●	UniversalConnector		●
	ODBC1	☐		●	UniversalConnector		●
	TCP	✓		●	UniversalConnector	MENU=UCEMAIL,FEP=1,RECEIVER=14,FIELDSET=SMS	●
	TCP1	☐		●	UniversalConnector	FIELDSET=XMLGPS,FEP=1,RECEIVER=1,LINE=1	●
*		☐		●			●

Line Driver properties

- To select properties for a Line Function, right-click in the Properties area of the Line Drivers form, and select "Properties". Select a Line Function from the dropdown menu. When you finish, click "**OK**".



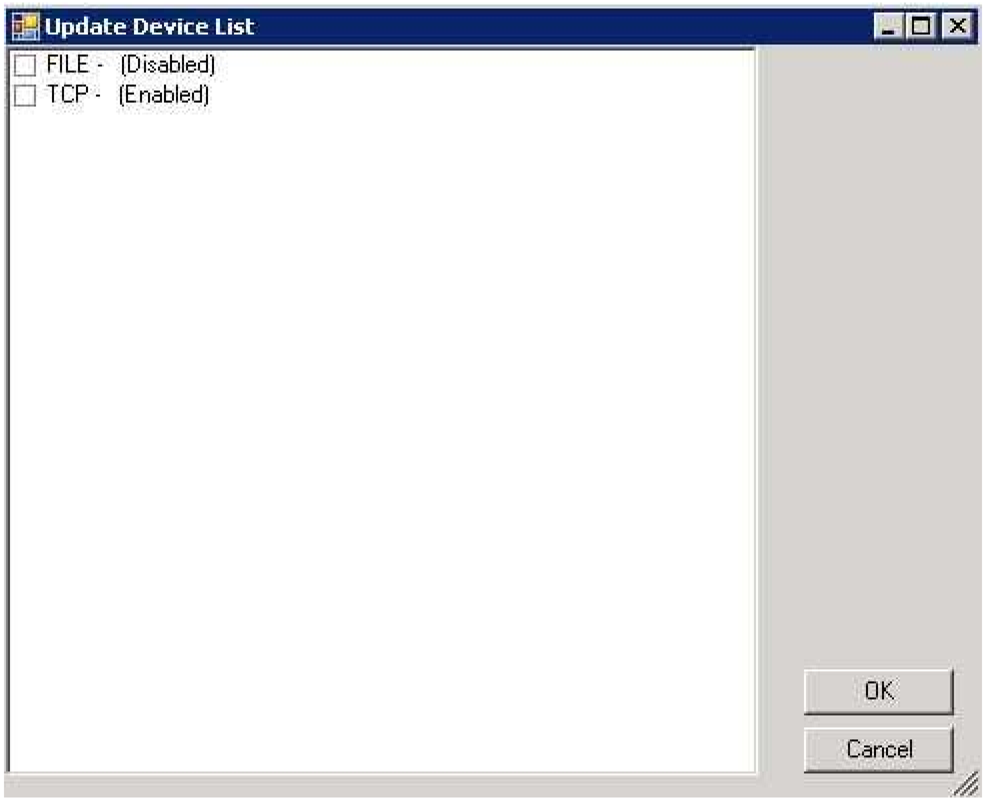
Function Properties

FieldSet	SEPARATOR
FEP	
Receiver	
Line	
Menu	
Lookup	

FieldSet
FieldSet to use for parsing the incoming signal

Line Drivers, Line Function Properties window

- To copy properties, right-click in the Properties area for a particular Line Function, and select "Copy To Properties". Then, select the Line Driver to which you want to copy the properties, and click "**OK**".



Line Drivers, Line Functions Copy To Update window

5. Click "**Save**".

Parameters

Use parameters to indicate specific preferences for MediaGateway 2 components. Parameters are typically assigned to Line Drivers, and can be edited at any time.

OpenVoice (In-Bound)

No parameters are required for OpenVoice In-Bound, but you may want to set the **MENU** parameter to a default menu. To specify a menu template or a created menu, right-click in the parameter field. When the "Function Properties" menu displays, enter the menu code you want in the "Menu" field, and click "**OK**".

Use the **LOOKUP** parameter if you want to provide multiple services on one phone line. For example, a Central Station may have a group of only four phone lines, but needs to provide services to ten different Dealers who all require unique services. With the LOOKUP function, the Dealer calling in can be routed to a specific menu based on DNIS, caller ID, or caller ID

Name. When using LOOKUP, the LOOKUP menu's Line Function must match the Line Function in the Line Driver configuration form.

Use the **OUTBOUNDPOOL** parameter if you want to transfer calls. The number you use for the OUTBOUNDPOOL parameter is arbitrary, but you must have outbound devices that match the pool number. Because this function randomly attempts to find a device with which to connect, the menu you use for the transfer should address the possibility of a failed attempt.

Use the **PRI=low-high** parameter to specify the alarm priority range you want to use for a specific line device. This parameter allows you to allocate a few lines for high-priority outbound calls. For example, PRI=1-5 handles priorities 1 through 5, while PRI=2 only handles priority 2.
Note: the system will not notify you regarding gaps in the priority range.

OpenVoice (Out-Bound)

The **RETRIES** option pools all pending outbound calls for alarm response, and puts them in a queue. The system then attempts the call several times before failing the Reverse Command back to Manitou. Then, it moves onto the next step in the Action Pattern.

SoloProtect

The parameters for the SoloProtect menu define the configuration you want for audio slices. **RECORDTIME** is the total elapsed time from the beginning of the call until the end. **SLICETIME** is the duration of each individual piece.
Each slice arrives as a separate signal into Manitou.

LineCheck

Use the **LINECHECK** function to check Receiver status by simulating a Contact ID alarm panel. LINECHECK dials out, and then delivers a Contact ID to the Receiver that checks its status.

Use the **CALLPOOL** parameter to set up a call pool using an arbitrary number. The CALLPOOL number you enter, however, must match the same parameter in the LOOKUP menu. You must also enter a Transmitter ID between one and four characters, a code, and a test zone.

Use the **INTERVAL** parameter to specify how often (in minutes) you want to test signals being sent in. This is for non-Manitou users only. The default value is 60 minutes.

Use the **ALLOWOV** parameter to approve OpenVoice usage for the line. Your options are 1 and 0. If you set the parameter to 1, you can use the line for OpenVoice.

Using LineCheck without the Auto-Client

Assign each line to a call pool, and set the interval you want. Include the numbers you want to call in the LOOKUP. The non-Manitou menu for LINECHECK delivers its responses via Contact ID to another Receiver (rather than by using the XML Receiver Driver).

PBX Assistant and Internal Line Configuration Parameters:

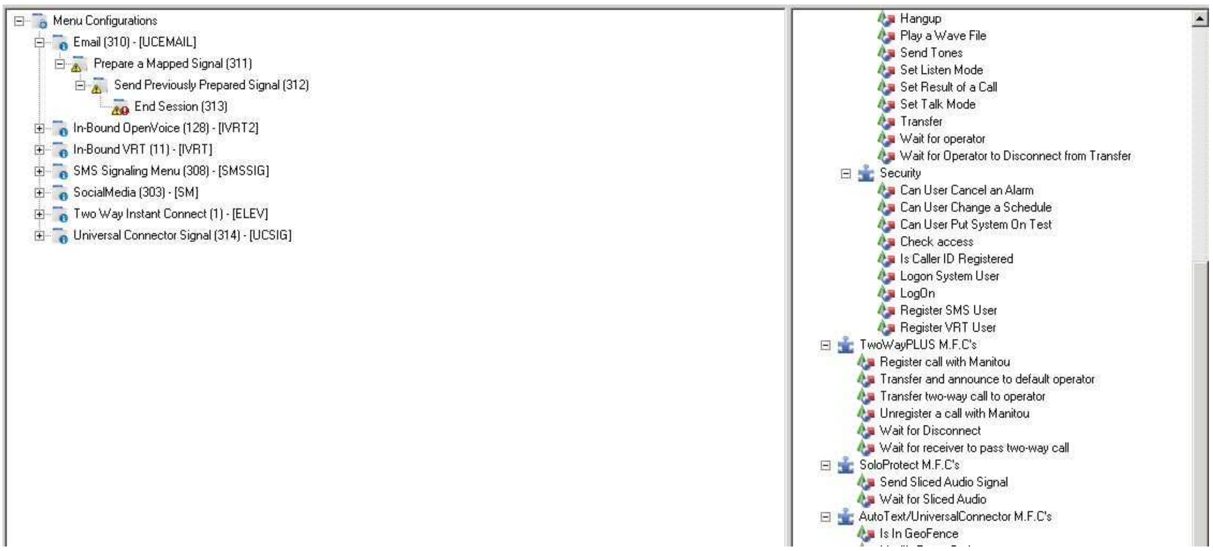
FEP - this is the FEP to which the Receiver is connected. When entering information into the form, separate multiple FEPs by a comma.
RECEIVER - this is the Receiver that receives the two-way signal. The Receiver number must match the Receiver you set up in the Supervisor Workstation.
LINE - this is the physical line of the Receiver.
TYPE - this indicates the type of two-way you are using. If you are using a Receiver, set this parameter to zero.
MENU - this indicates the type of menu you want to use with two-way.
LINEID - this reference field is used to group line transfers together. It must match the LINEID for the Outbound call.
USEDNIS - this option indicates that DNIS should be sent to Receivers capable of accepting it. Specify that it is for a Surgard Receiver. If the parameter is set to 0, the system sends a message between the first and second ring to pass the caller ID information without the DNIS. If set to 1, the system sends DTMF tones to the Receiver in the following format: <ANI>#<DNIS>#.
LINEID - this option indicates to the Operator which line you want to use as a phone line. Select "Internal Transfer Line" as the Line Function. The LINEID must match the LINEID for the two-way configuration.
STATIONID - this option indicates the station port you want to use. Enter "Line to Receiver" as the description. Select "Internal Transfer Line" as the Line Function. Set this parameter to 1.

AutoText

The **TIMEOUT** parameter is the length of time (in minutes) OpenVoice waits for a response before failing.

MediaGateway 2 Menu Items

Menu items are the core components of the MediaGateway 2. The MediaGateway 2 comes with a number of pre-defined menu templates. You can edit these pre-defined menus to meet your specific needs. You can also create new menus. The MediaGateway 2 provides unique menu structures for almost any situation.


Menu Configurations
Email (310) - [UCEMAIL]
Prepare a Mapped Signal (311)
Send Previously Prepared Signal (312)
End Session (313)
In-Bound OpenVoice (128) - [IVRT2]
In-Bound VRT (11) - [IVRT]
SMS Signaling Menu (308) - [SMSSIG]
SocialMedia (303) - [SM]
Two Way Instant Connect (1) - [ELEV]
Universal Connector Signal (314) - [UCSIG]
Hangup
Play a Wave File
Send Tones
Set Listen Mode
Set Result of a Call
Set Talk Mode
Transfer
Wait for operator
Wait for Operator to Disconnect from Transfer
Security
Can User Cancel an Alarm
Can User Change a Schedule
Can User Put System On Test
Check access
Is Caller ID Registered
Logon System User
LogOn
Register SMS User
Register VRT User
TwoWayPLUS M.F.C's
Register call with Manitou
Transfer and announce to default operator
Transfer two-way call to operator
Unregister a call with Manitou
Wait for Disconnect
Wait for receiver to pass two-way call
SoloProtect M.F.C's
Send Sliced Audio Signal
Wait for Sliced Audio
AutoText/UniversalConnector M.F.C's
Is In GeoFence

Menu Items

The MediaGateway 2 menu items window is divided into two sections. Manitou Functional Components (M.F.C.s) display on the right side of the window, and menu structures display on the left. Menu structures are composed of multiple M.F.C.s strung together and organized into a hierarchical procedure. A menu structure is a complete functional system designed to handle a specific workflow problem (for e.g., handling inbound user calls).

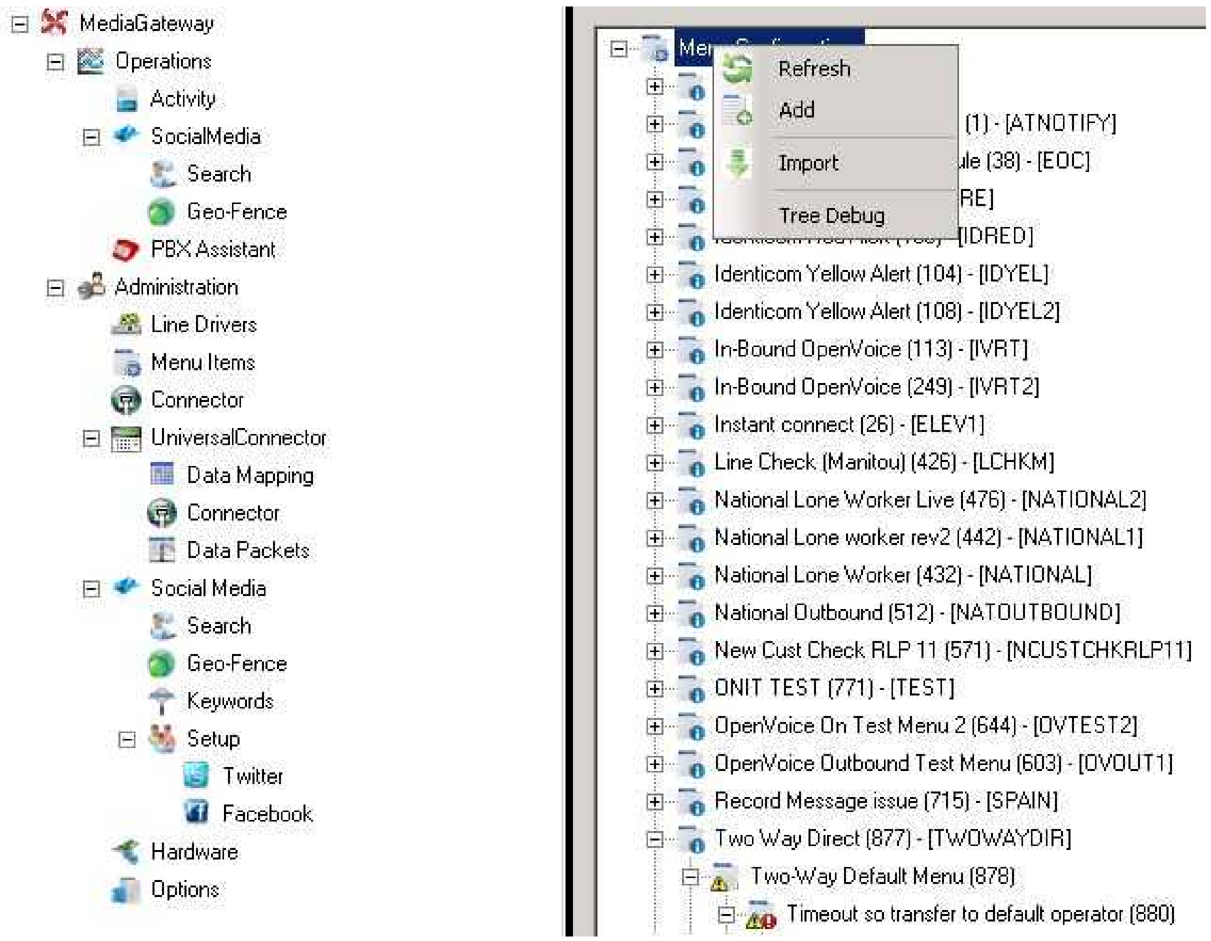
Importing and Exporting Menus

Your ability to import and export menu structures provides you with a quick and flexible way to distribute, update, and copy menus.

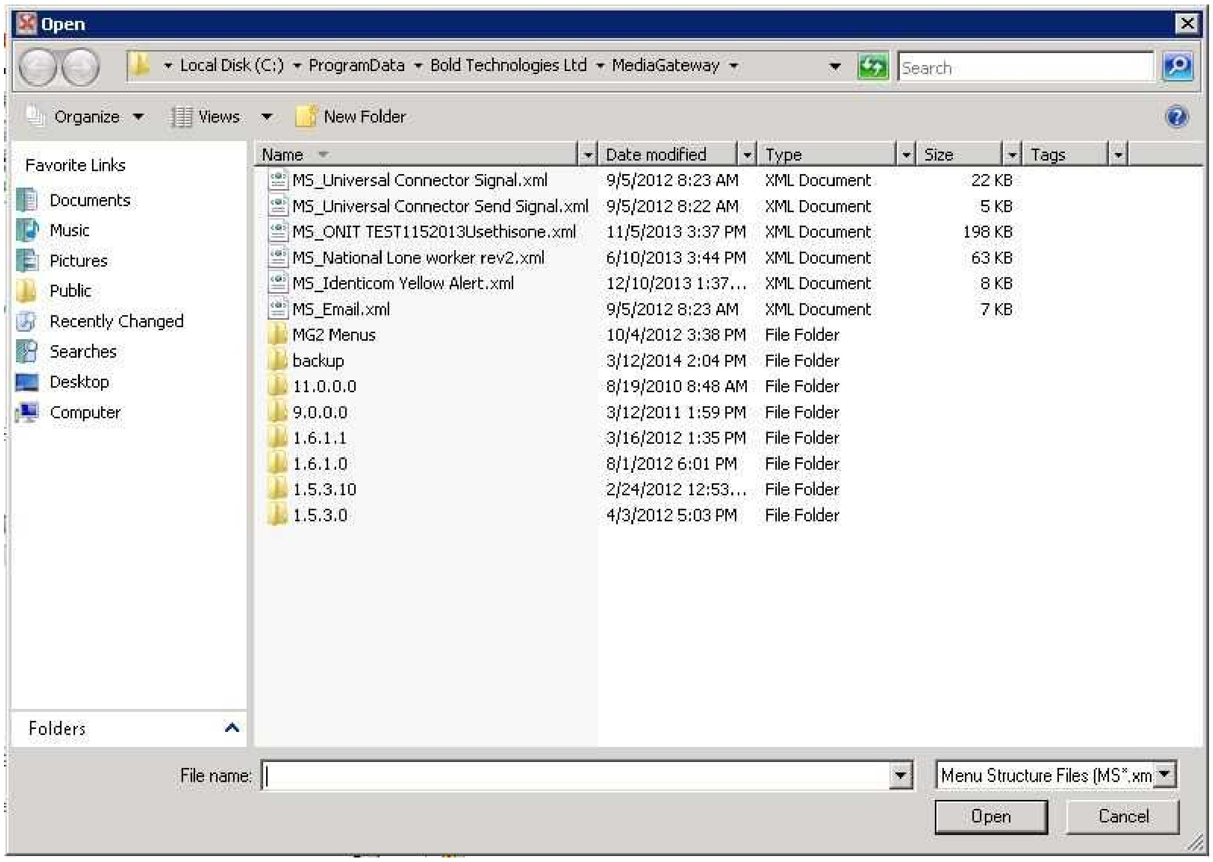
Importing Menus

Perform the following steps to import menus into the MediaGateway 2:

1. Select **"**Menu Items" under "Administration" from the the Navigation Tree.
2. Right-click on "Menu Configurations" at the top of the window, and then select "Import" as shown in the following screenshot:


MediaGateway
Operations
Activity
SocialMedia
Search
Geo-Fence
PBX Assistant
Administration
Line Drivers
Menu Items
Connector
UniversalConnector
Data Mapping
Connector
Data Packets
Social Media
Search
Geo-Fence
Keywords
Setup
Twitter
Facebook
Hardware
Options
Refresh
Add
Import
Tree Debug
(1) - [ATNOTIFY]
ule (38) - [EOC]
RE]
[IDRED]
Identicom Yellow Alert (104) - [IDYEL]
Identicom Yellow Alert (108) - [IDYEL2]
In-Bound OpenVoice (113) - [IVRT]
In-Bound OpenVoice (249) - [IVRT2]
Instant connect (26) - [ELEV1]
Line Check (Manitou) (426) - [LCHKM]
National Lone Worker Live (476) - [NATIONAL2]
National Lone worker rev2 (442) - [NATIONAL1]
National Lone Worker (432) - [NATIONAL]
National Outbound (512) - [NATOUTBOUND]
New Cust Check RLP 11 (571) - [NCUSTCHKRLP11]
ONIT TEST (771) - [TEST]
OpenVoice On Test Menu 2 (644) - [OVTEST2]
OpenVoice Outbound Test Menu (603) - [OVOUT1]
Record Message issue (715) - [SPAIN]
Two Way Direct (877) - [TWOWAYDIR]
Two-Way Default Menu (878)
Timeout so transfer to default operator (880)

Result: the "Open" window displays as shown in the following screenshot:



Name	Date modified	Type	Size	Tags
MS_Universal Connector Signal.xml	9/5/2012 8:23 AM	XML Document	22 KB	
MS_Universal Connector Send Signal.xml	9/5/2012 8:22 AM	XML Document	5 KB	
MS_ONIT TEST1152013Usethisone.xml	11/5/2013 3:37 PM	XML Document	198 KB	
MS_National Lone worker rev2.xml	6/10/2013 3:44 PM	XML Document	63 KB	
MS_Identicom Yellow Alert.xml	12/10/2013 1:37...	XML Document	8 KB	
MS_Email.xml	9/5/2012 8:23 AM	XML Document	7 KB	
MG2 Menus	10/4/2012 3:38 PM	File Folder		
backup	3/12/2014 2:04 PM	File Folder		
11.0.0.0	8/19/2010 8:48 AM	File Folder		
9.0.0.0	3/12/2011 1:59 PM	File Folder		
1.6.1.1	3/16/2012 1:35 PM	File Folder		
1.6.1.0	8/1/2012 6:01 PM	File Folder		
1.5.3.10	2/24/2012 12:53...	File Folder		
1.5.3.0	4/3/2012 5:03 PM	File Folder		

3. Navigate to the XML file you want to import.
4. Select the file, and then click "**OK**".

Result: MediaGateway 2 imports the menu you selected. If the menu already exists in the MediaGateway 2, however, the system will prompt you to overwrite it.

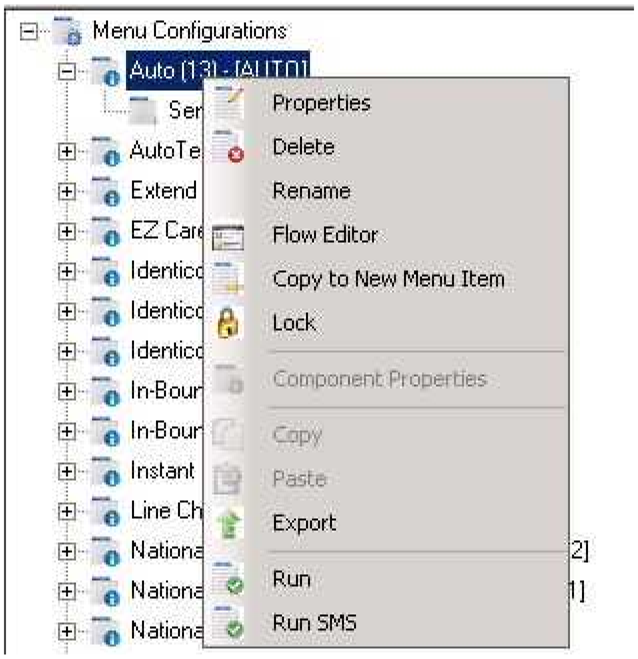
Exporting Menus

You can export and save menus as XML files. This functions as a backup and allows for future access (i.e., to use when building a new menu based on an existing menu structure).

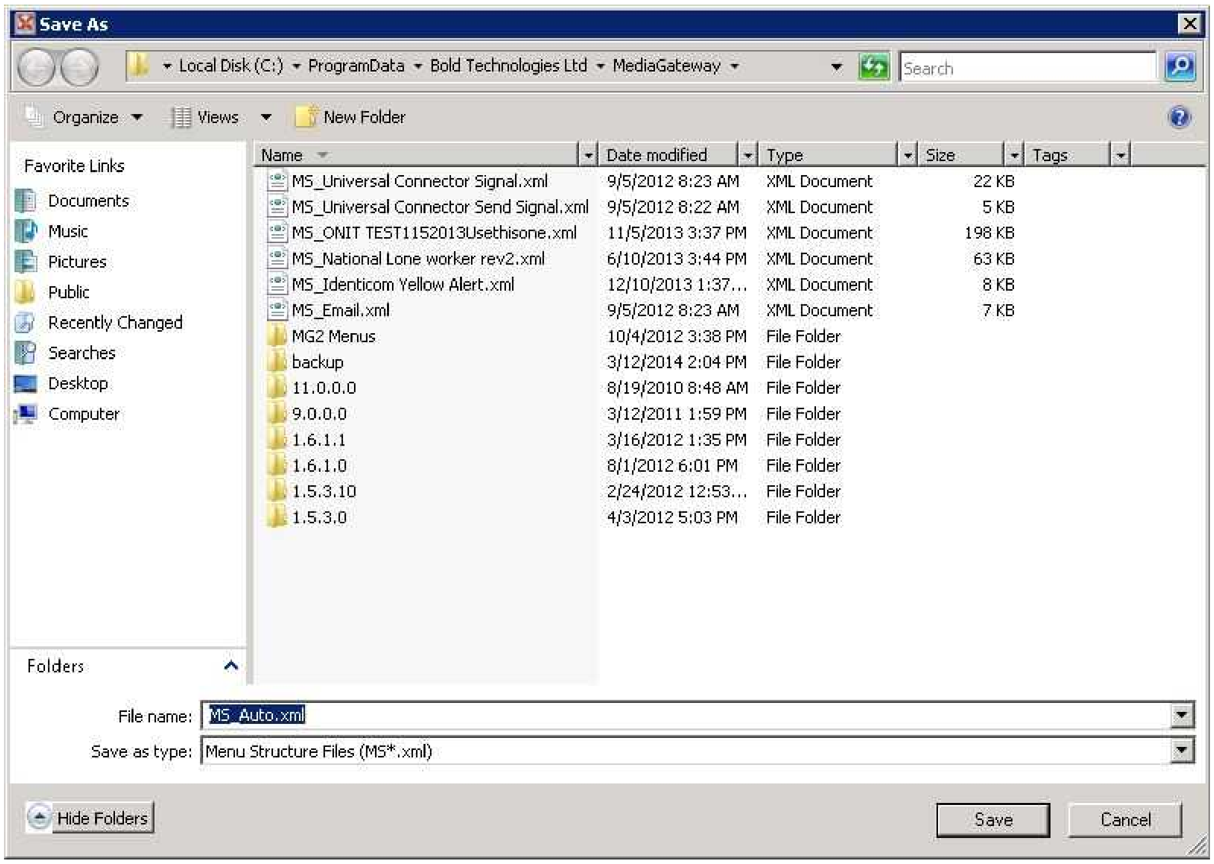
Perform the following steps to export a MediaGateway 2 menu:

1. Select **"**Menu Items" under "Administration" from the the Navigation Tree.
2. Right-click on the top node of the menu you want to export.

Result: the following menu displays:



3. Select "Export".
Result: the "Save As" window displays as shown in the following screenshot:



Name	Date modified	Type	Size	Tags
MS_Universal Connector Signal.xml	9/5/2012 8:23 AM	XML Document	22 KB	
MS_Universal Connector Send Signal.xml	9/5/2012 8:22 AM	XML Document	5 KB	
MS_ONIT TEST1152013Usethisone.xml	11/5/2013 3:37 PM	XML Document	198 KB	
MS_National Lone worker rev2.xml	6/10/2013 3:44 PM	XML Document	63 KB	
MS_Identicom Yellow Alert.xml	12/10/2013 1:37...	XML Document	8 KB	
MS_Email.xml	9/5/2012 8:23 AM	XML Document	7 KB	
MG2 Menus	10/4/2012 3:38 PM	File Folder		
backup	3/12/2014 2:04 PM	File Folder		
11.0.0.0	8/19/2010 8:48 AM	File Folder		
9.0.0.0	3/12/2011 1:59 PM	File Folder		
1.6.1.1	3/16/2012 1:35 PM	File Folder		
1.6.1.0	8/1/2012 6:01 PM	File Folder		
1.5.3.10	2/24/2012 12:53...	File Folder		
1.5.3.0	4/3/2012 5:03 PM	File Folder		

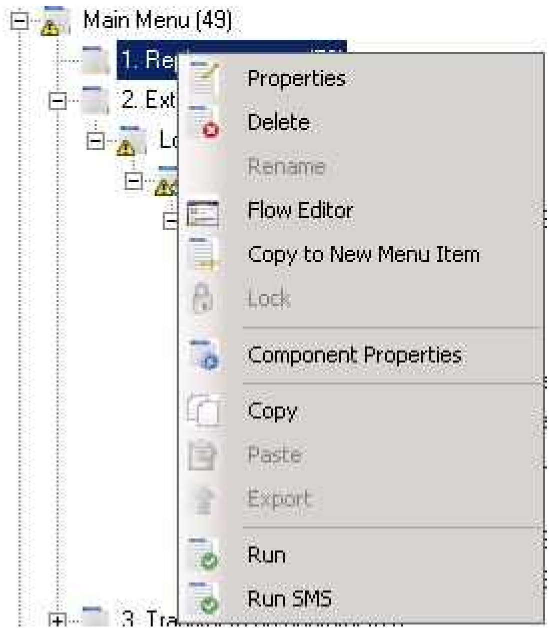
File name: MS_Auto.xml
Save as type: Menu Structure Files (MS*.xml)

4. Navigate to the XML file you want to export.
5. Select the file, and then click **"Save"**.

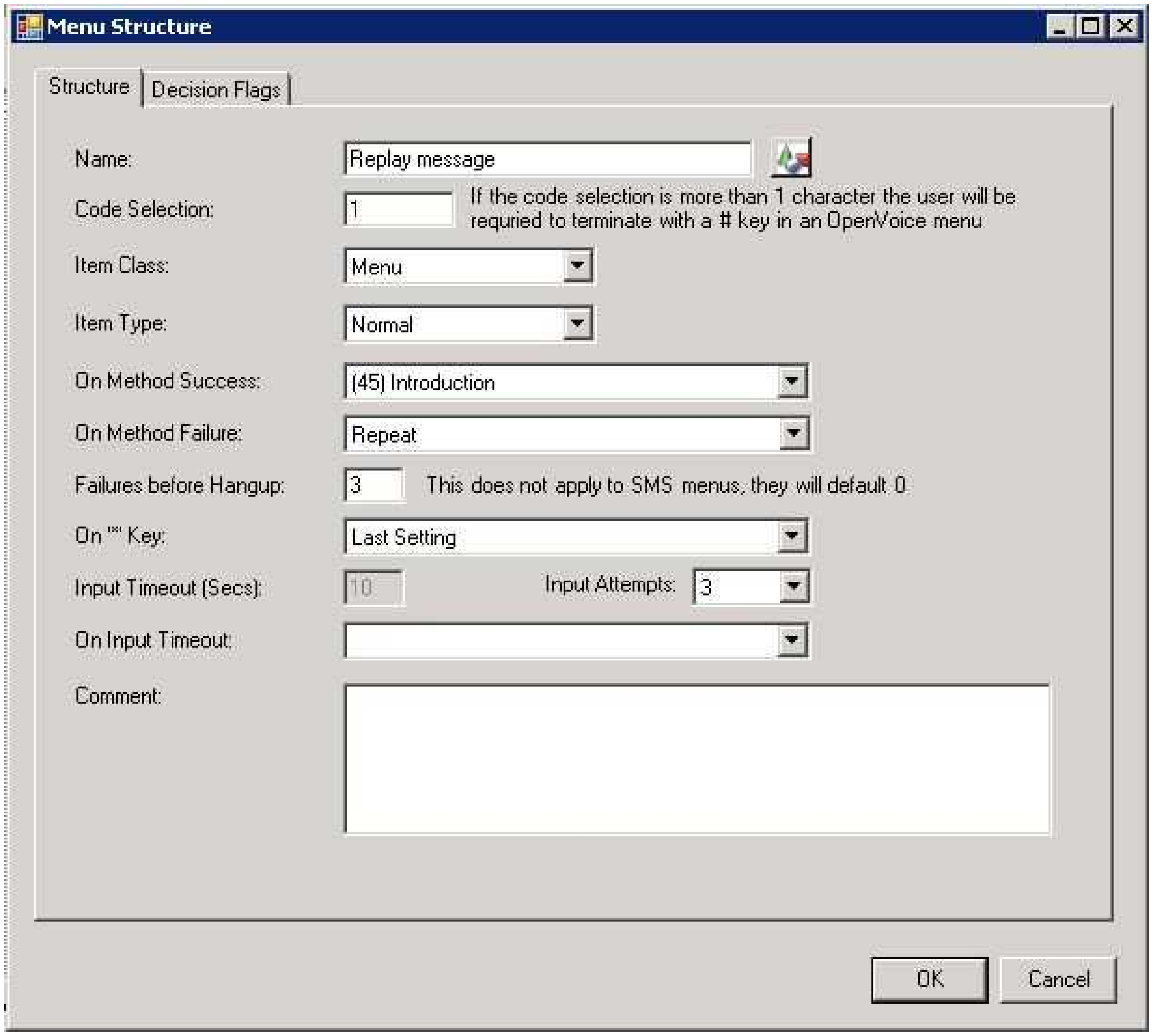
Result: MediaGateway 2 exports the menu you selected. If the menu already exists in the destination folder, however, the system will prompt you to overwrite it.

Editing Menu Properties

You can edit menu properties by right-clicking the menu item you want to edit in "Menu Items". Then, select the "Properties" option as shown in the following screenshot:



Result: the "Menu Structure" window displays as shown in the following screenshot:



Name: Replay message

Code Selection: 1 If the code selection is more than 1 character the user will be requried to terminate with a # key in an OpenVoice menu

Item Class: Menu

Item Type: Normal

On Method Success: (45) Introduction

On Method Failure: Repeat

Failures before Hangup: 3 This does not apply to SMS menus, they will default 0

On "*" Key: Last Setting

Input Timeout (Secs): 10 Input Attempts: 3

On Input Timeout:

Comment:

Name Field

In the Menu Structure example above, the name of the menu is "Replay message." The menu name is not conveyed to the user. It is only used for internal purposes when you are building a menu.

Code Selection Field

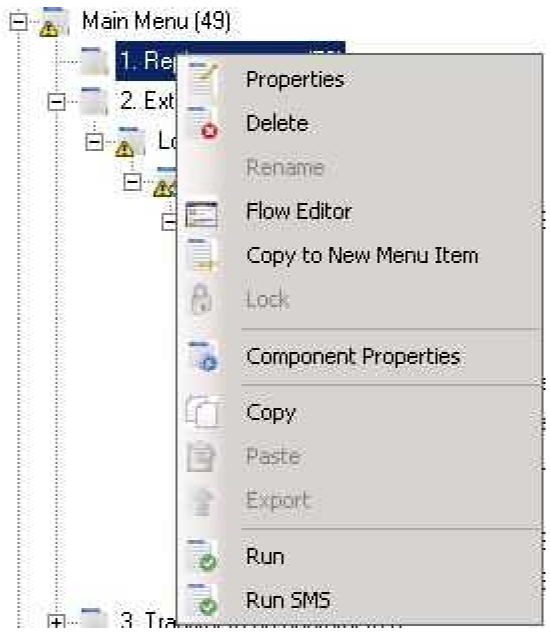
Use the "Code Selection:" field with menu items that require numeric prompts (for e.g., "Press 1 for English"). Enter the number with which you want the user to be prompted.

In the "Replay Message" example above, the menu is set to play the "Introduction" message if the user presses "1" on his phone keypad when prompted. If he fails to press "1", the message repeats. The menu is set to replay the message three times before hanging up. Pressing "*" will also cause the the message to repeat because the "On * Key:" field is set to "Last Setting".

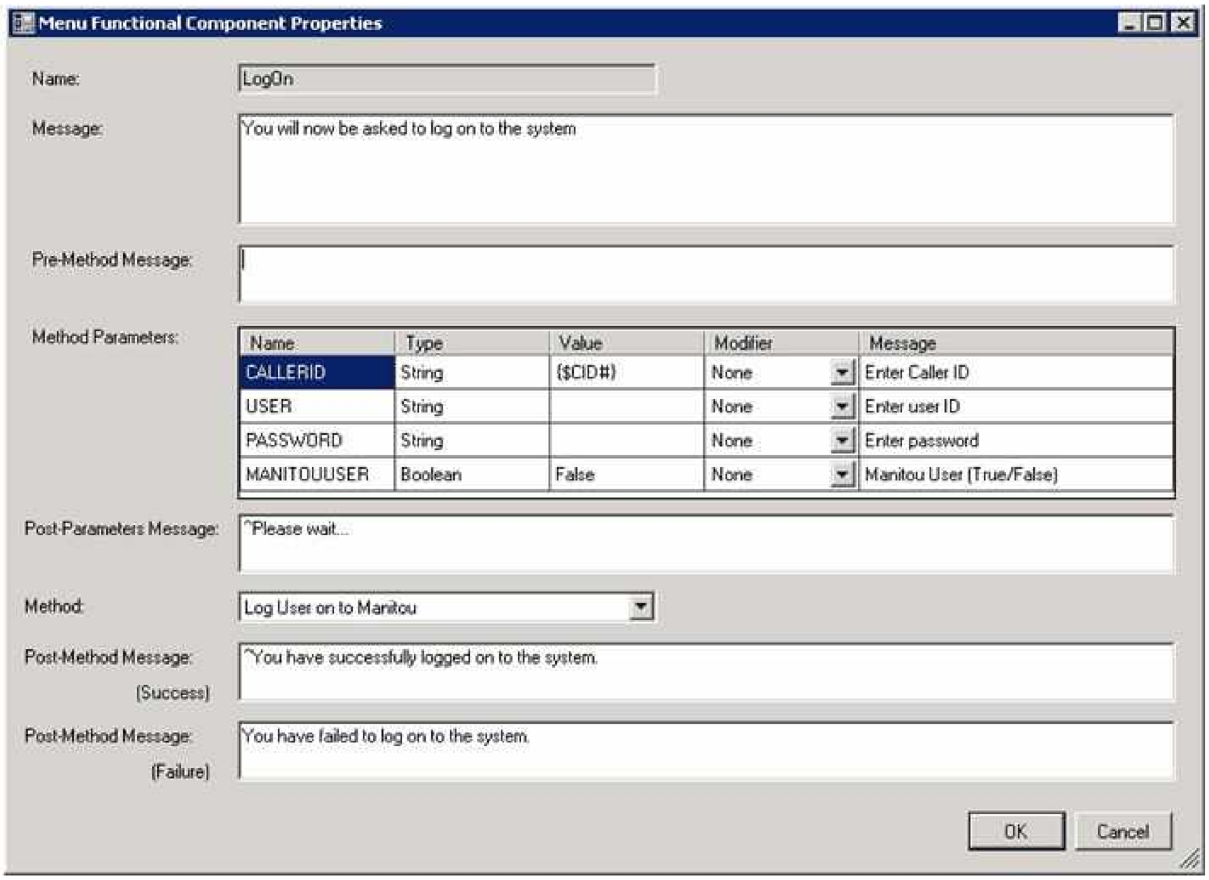
You can also define Failure and Timeout parameters (for e.g., the number of seconds that must pass before disconnecting, or how many failures the user can have before the connection is terminated).

Component Properties

Use the Menu Functional Component Properties form to define text-to-voice messages, and to add and define method parameters. To access the form, right-click on the menu item you want, and then select "Component Properties" as shown in the following screenshot:



Result: the Menu Functional Component Properties window displays as shown in the following screenshot:



Menu Functional Component Properties

Name:	LogOn
Message:	You will now be asked to log on to the system
Pre-Method Message:	

Method Parameters:

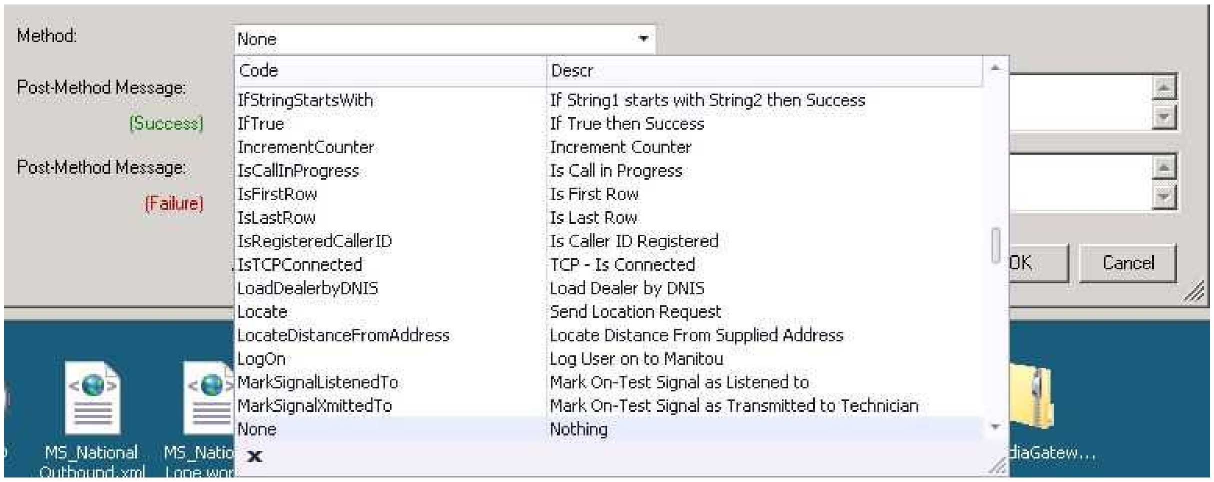
Name	Type	Value	Modifier	Message
CALLERID	String	{$CID#}	None	Enter Caller ID
USER	String		None	Enter user ID
PASSWORD	String		None	Enter password
MANITOUUSER	Boolean	False	None	Manitou User (True/False)

Post-Parameters Message:	^Please wait...
Method:	Log User on to Manitou
Post-Method Message: (Success)	^You have successfully logged on to the system.
Post-Method Message: (Failure)	You have failed to log on to the system.

Component Properties

In the example above for the LogOn menu:

- The "Message:" field displays what the user hears as he begins the menu item process. A voice will say: "You will now be asked to log on to the system".
- The "Pre-Method Message:" field contains a message used as a pre-method prompt (for e.g., "Please wait").
- The "Method Parameters:" fields contain user prompts authenticated by Manitou (for e. g., user ID and password). To add a new method, select the one you want from the "Method:" dropdown list as shown in the following screenshot:



Method:

None

Post-Method Message:

(Success)

Post-Method Message:

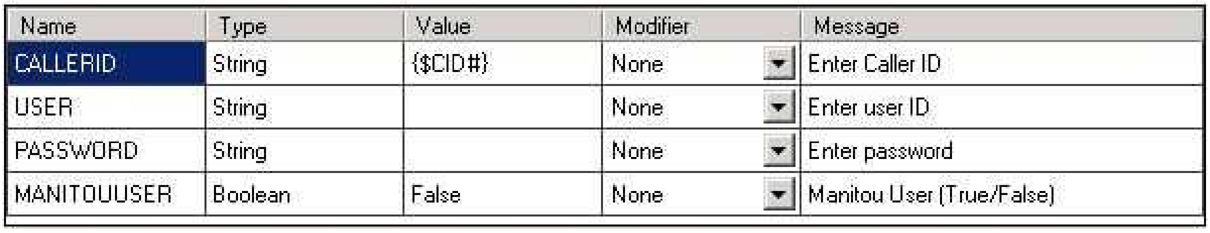
(Failure)

Code	Descr
IfStringStartsWith	If String1 starts with String2 then Success
IfTrue	If True then Success
IncrementCounter	Increment Counter
IsCallInProgress	Is Call in Progress
IsFirstRow	Is First Row
IsLastRow	Is Last Row
IsRegisteredCallerID	Is Caller ID Registered
IsTCPConnected	TCP - Is Connected
LoadDealerbyDNIS	Load Dealer by DNIS
Locate	Send Location Request
LocateDistanceFromAddress	Locate Distance From Supplied Address
LogOn	Log User on to Manitou
MarkSignalListenedTo	Mark On-Test Signal as Listened to
MarkSignalXmittedTo	Mark On-Test Signal as Transmitted to Technician
None	Nothing

In the LogOn example above, the method is logging a user into Manitou. Once you select a parameter, the system automatically adds it to the "Method Parameters:" field list.

Method Parameters

The following is an example of a "Method Parameter" table from the Menu Functional Component Properties form:

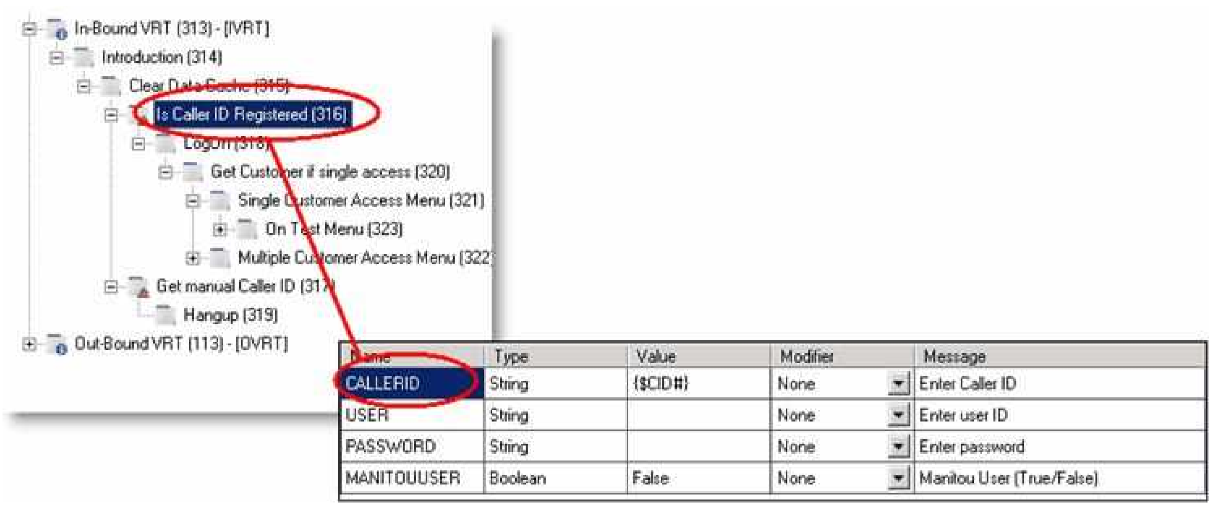


Name	Type	Value	Modifier	Message
CALLERID	String	{$CID#}	None	Enter Caller ID
USER	String		None	Enter user ID
PASSWORD	String		None	Enter password
MANITOUUSER	Boolean	False	None	Manitou User (True/False)

Method Parameters

The "Name:" and "Type:" fields are read-only and cannot be edited.
The "Value:" column is editable and contains the information the MediaGateway 2 looks for in order to continue with the next step in the method.

In the following example for the "CALLERID" method, the MediaGateway 2 attempts to verify the caller ID number, and determine whether it is registered. In fact, the following screenshot indicates that the menu item prior to logging on checks to see whether the caller ID is registered:

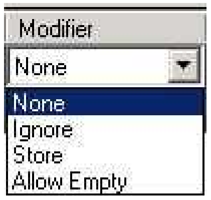

In-Bound VRT (313) - [IVRT]
Introduction (314)
Clear Data Cache (315)
Is Caller ID Registered (316)
LogOn (318)
Get Customer if single access (320)
Single Customer Access Menu (321)
On Test Menu (323)
Multiple Customer Access Menu (322)
Get manual Caller ID (317)
Hangup (319)
Out-Bound VRT (113) - [OVRT]

Name	Type	Value	Modifier	Message
CALLERID	String	{$CID#}	None	Enter Caller ID
USER	String		None	Enter user ID
PASSWORD	String		None	Enter password
MANITOUUSER	Boolean	False	None	Manitou User (True/False)

Previous Menu Item

If the caller ID is not registered, the system prompts the user to enter his caller ID as part of the LogOn method process.

As shown in the following screenshot, the "Modifier" column contains a dropdown menu that indicates whether you want the system to store the information or ignore it:



In the Parameters "Message:" field, enter the text you want to play to the user over the phone or to display via text message. In the above example, the user hears a message that says: "Enter Caller ID". After the user enters a caller ID, the system prompts him for a user ID.

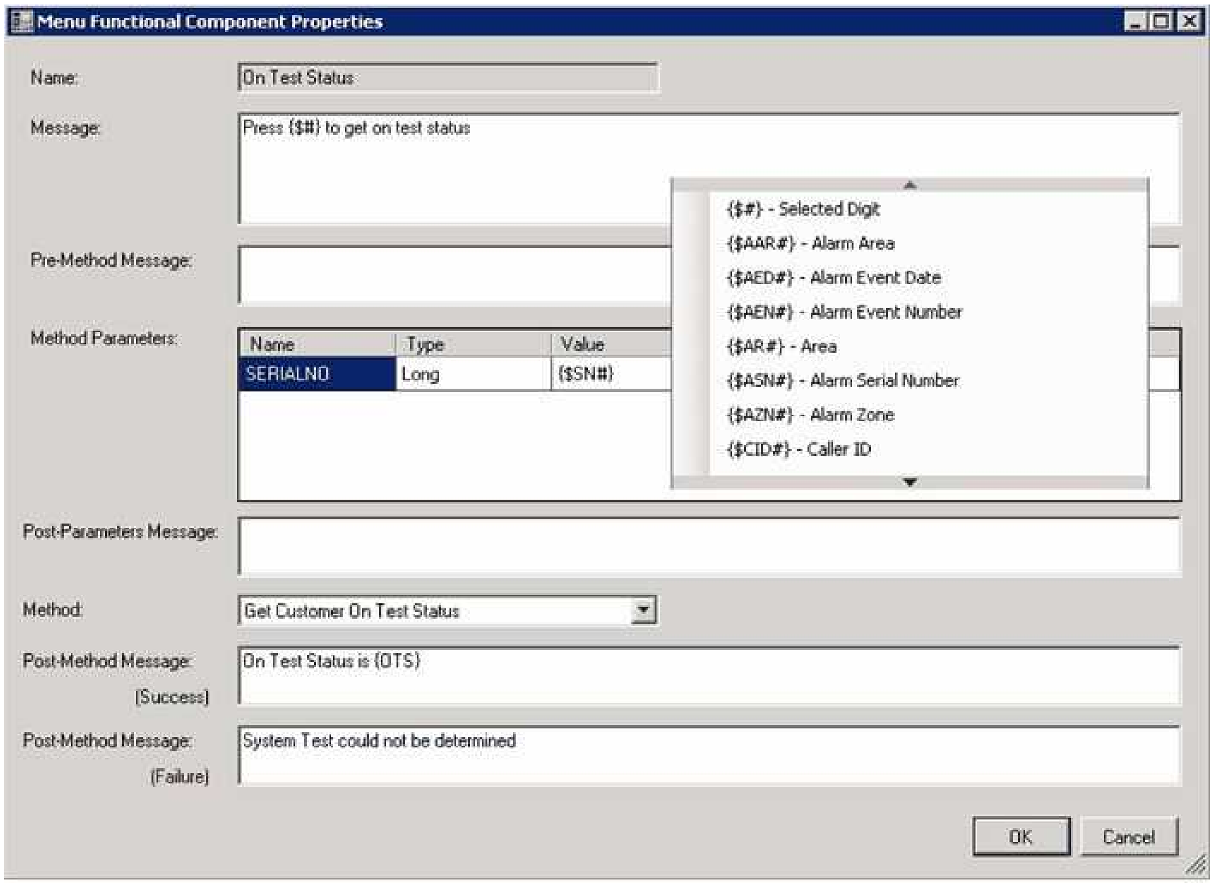
When the method process is finished, the "Post-Method Message" plays or displays to the user. An example of a Post-Method Message is: "Please wait while your information is verified."

As shown in the screenshot of the Menu Functional Component Properties window above, you may want to include a "^" character prior to message text. This indicates that only phone users hear this message. This option exists so that pay-per-message SMS users do not incur charges for unnecessary messages.

Finally, if all method parameters have been successfully met, the system plays or displays the Success Message. An example of a Success Message is: "You have successfully logged on to the system." If the logon attempt fails, however, users hear or receive a Failure message such as "You have failed to log on to the system."

Using Values

In the "Message:", "Pre-Method Message:", "Post-Parameters Message:", and "Post-Method Message:" fields, you can select values that change according to your information. In the example below, a technician is putting a Customer On Test, and is prompted to enter a numeric value (defined in the Menu Structure form) to begin the process:



Name: On Test Status

Message: Press {$#} to get on test status

Pre-Method Message:

Method Parameters:

Name	Type	Value
SERIALNO	Long	{$SN#}

{$#} - Selected Digit
{$AAR#} - Alarm Area
{$AED#} - Alarm Event Date
{$AEN#} - Alarm Event Number
{$AR#} - Area
{$ASN#} - Alarm Serial Number
{$AZN#} - Alarm Zone
{$CID#} - Caller ID

Post-Parameters Message:

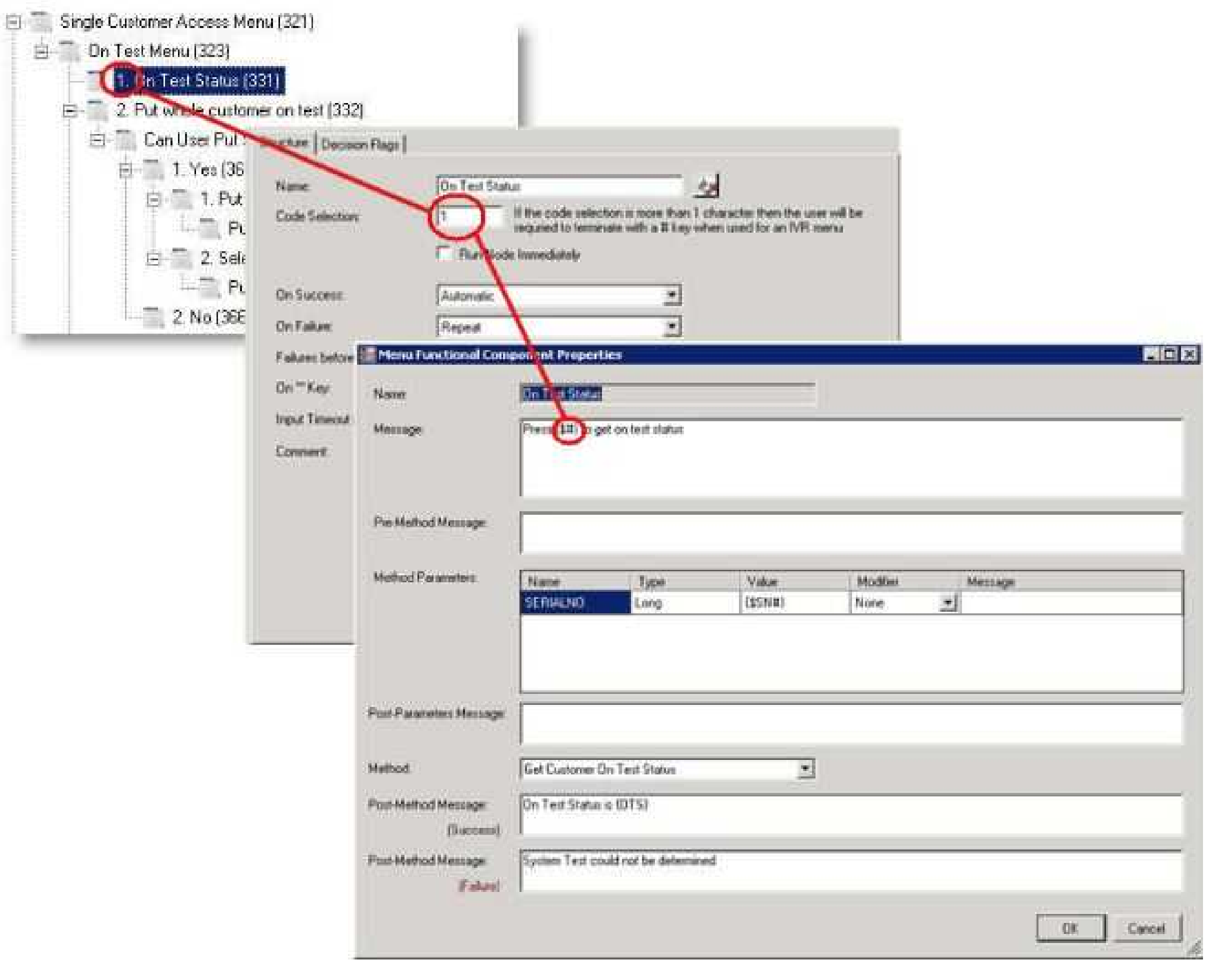
Method: Get Customer On Test Status

Post-Method Message: (Success) On Test Status is {OTS}

Post-Method Message: (Failure) System Test could not be determined

Values

Right-clicking in any of the aforementioned fields displays the "Values" dialog, where you can select these values. In the example above, the {$#} value was selected from the Values dialog. The example below shows how the values are connected:



Single Customer Access Menu (321)
On Test Menu (323)
1. On Test Status (331)
2. Put whole customer on test (332)
Can User Put
1. Yes (36
1. Put
Pu
2. Sel
Pu
2. No (36

Name:	On Test Status
Code Selection:	1
	If the code selection is more than 1 character then the user will be required to terminate with a # key when used for an IVR menu
	Run Code Immediately
On Success:	Automatic
On Failure:	Repeat
Failures before	
On * Key:	
Input Timeout:	
Comment:	

Menu Functional Component Properties

Name	On Test Status
Message	Press {#} to get on test status
Pre-Method Message:	
Method Parameters:	

Name	Type	Value	Modifier	Message
SERIALNO	Long	{$SN#}	None	

Post-Parameters Message:	
Method:	Get Customer On Test Status
Post-Method Message: (Success)	On Test Status is {OTS}
Post-Method Message: (Failure)	System Test could not be determined

Associated Values

The value was first indicated in the Menu Structure form by entering "1" into the "Code Selection:" field, which displays as "1. On Test Status" in the actual menu. In the "Menu Functional Component Properties" form, the message displays as: "Press 1 to get On Test Status." Then, press "1" to receive the On Test Status (indicated as "{OTS}" in the successful Post-Method Message).

M.F.C.s

For ease of reference, M.F.C.s have been divided into Standard and Module categories.
Standard M.F.C.s

Module M.F.C.s:
TwoWay Plus M.F.C.s
SoloProtect M.F.C.s
AutoText/UniversalConnector M.F.C.s

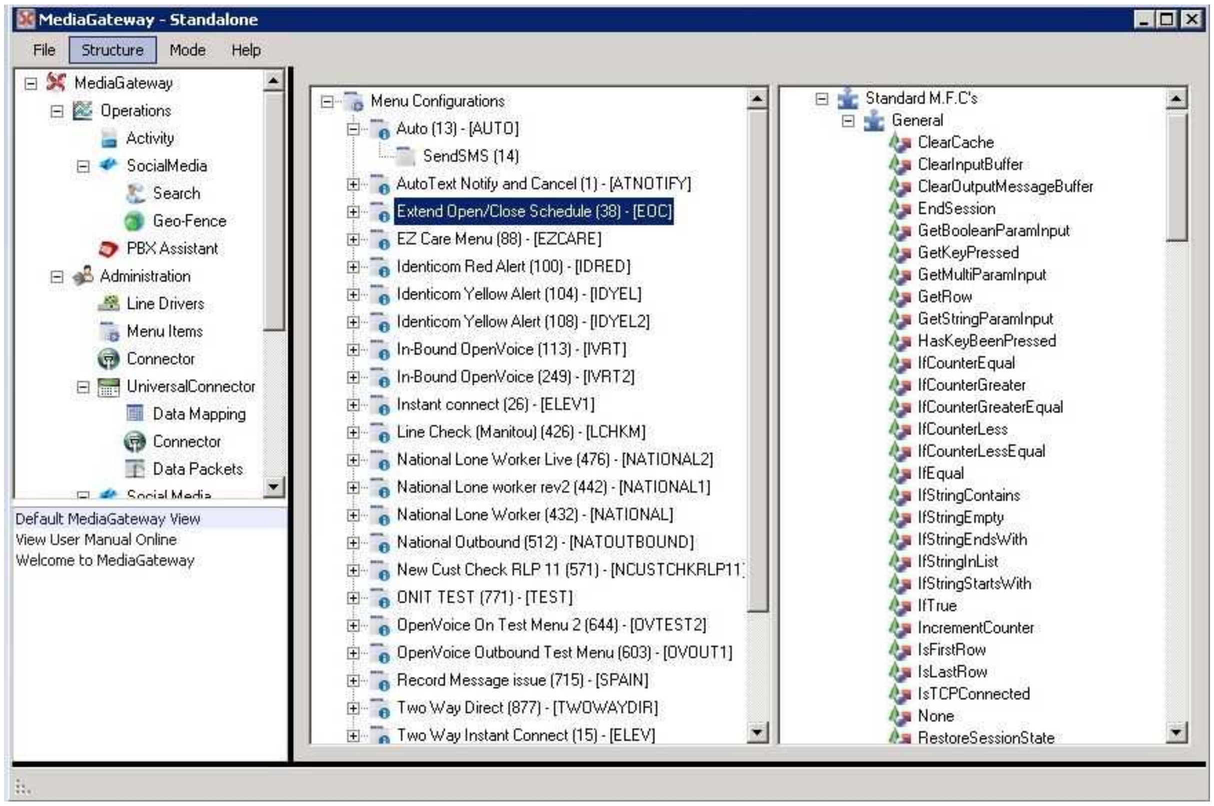
SocialMedia M.F.C.s

For a more comprehensive list, please refer to Appendix A - M.F.C. Full List.

Adding an M.F.C. to an Existing Menu Structure

Perform the following steps to add an M.F.C. to existing menu structure:

1. In the MediaGateway 2, click "Menu Items" under Administration in the Navigation Tree.
Result: the following window displays:



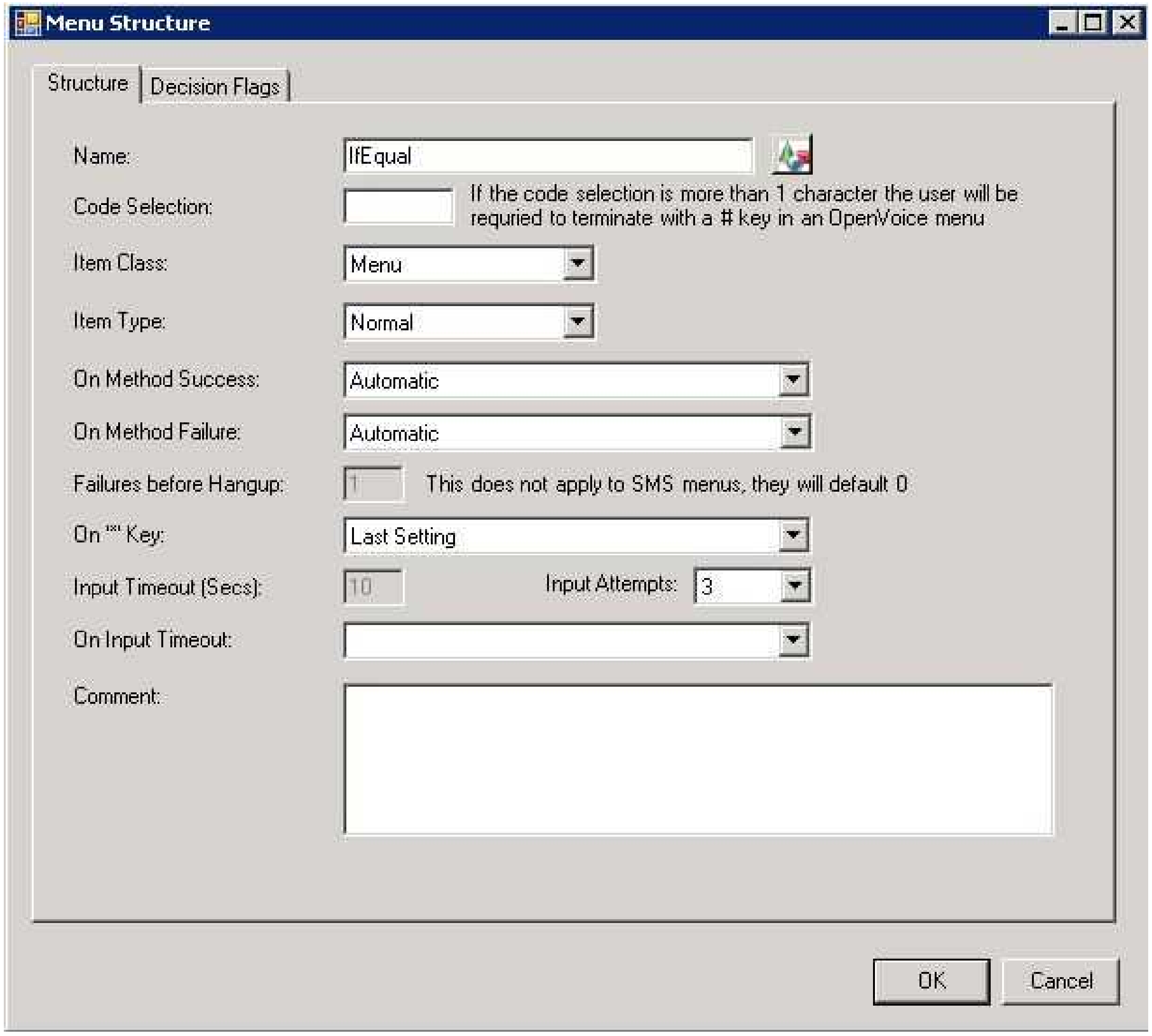
- MediaGateway
  - Operations
    - Activity
    - SocialMedia
      - Search
      - Geo-Fence
    - PBX Assistant
  - Administration
    - Line Drivers
    - Menu Items
    - Connector
    - UniversalConnector
      - Data Mapping
      - Connector
      - Data Packets
    - Social Media

Default MediaGateway View
View User Manual Online
Welcome to MediaGateway

- Menu Configurations
  - Auto (13) - [AUTO]
    - SendSMS (14)
  - AutoText Notify and Cancel (1) - [ATNOTIFY]
  - Extend Open/Close Schedule (38) - [EOC]
  - EZ Care Menu (88) - [EZCARE]
  - Identicom Red Alert (100) - [IDRED]
  - Identicom Yellow Alert (104) - [IDYEL]
  - Identicom Yellow Alert (108) - [IDYEL2]
  - In-Bound OpenVoice (113) - [IVRT]
  - In-Bound OpenVoice (249) - [IVRT2]
  - Instant connect (26) - [ELEV1]
  - Line Check (Manitou) (426) - [LCHKM]
  - National Lone Worker Live (476) - [NATIONAL2]
  - National Lone worker rev2 (442) - [NATIONAL1]
  - National Lone Worker (432) - [NATIONAL]
  - National Outbound (512) - [NATOUTBOUND]
  - New Cust Check RLP 11 (571) - [NCUSTCHKRLP11
  - ONIT TEST (771) - [TEST]
  - OpenVoice On Test Menu 2 (644) - [OVTEST2]
  - OpenVoice Outbound Test Menu (603) - [OVOUT1]
  - Record Message issue (715) - [SPAIN]
  - Two Way Direct (877) - [TWOWAYDIR]
  - Two Way Instant Connect (15) - [ELEV]

- Standard M.F.C's
  - General
    - ClearCache
    - ClearInputBuffer
    - ClearOutputMessageBuffer
    - EndSession
    - GetBooleanParamInput
    - GetKeyPressed
    - GetMultiParamInput
    - GetRow
    - GetStringParamInput
    - HasKeyBeenPressed
    - IfCounterEqual
    - IfCounterGreater
    - IfCounterGreaterEqual
    - IfCounterLess
    - IfCounterLessEqual
    - IfEqual
    - IfStringContains
    - IfStringEmpty
    - IfStringEndsWith
    - IfStringInList
    - IfStringStartsWith
    - IfTrue
    - IncrementCounter
    - IsFirstRow
    - IsLastRow
    - IsTCPConnected
    - None
    - RestoreSessionState

2. Expand the menu structure to which you want to add the new M.F.C. in the center window.
3. Select to highlight the M.F.C. you want to add from the window on the right.
4. Drag and drop the M.F.C. from the window on the right into the destination you want in the center window.
Result: the "Menu Structure" form displays as shown in the following screenshot:



Name: IfEqual

Code Selection:

If the code selection is more than 1 character the user will be requried to terminate with a # key in an OpenVoice menu

Item Class: Menu

Item Type: Normal

On Method Success: Automatic

On Method Failure: Automatic

Failures before Hangup: 1

This does not apply to SMS menus, they will default 0

On "*" Key: Last Setting

Input Timeout (Secs): 10

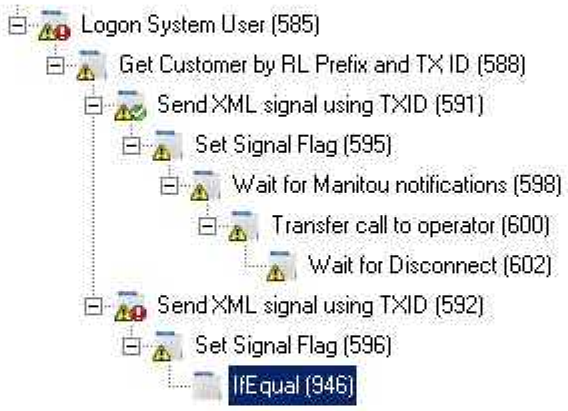
Input Attempts: 3

On Input Timeout:

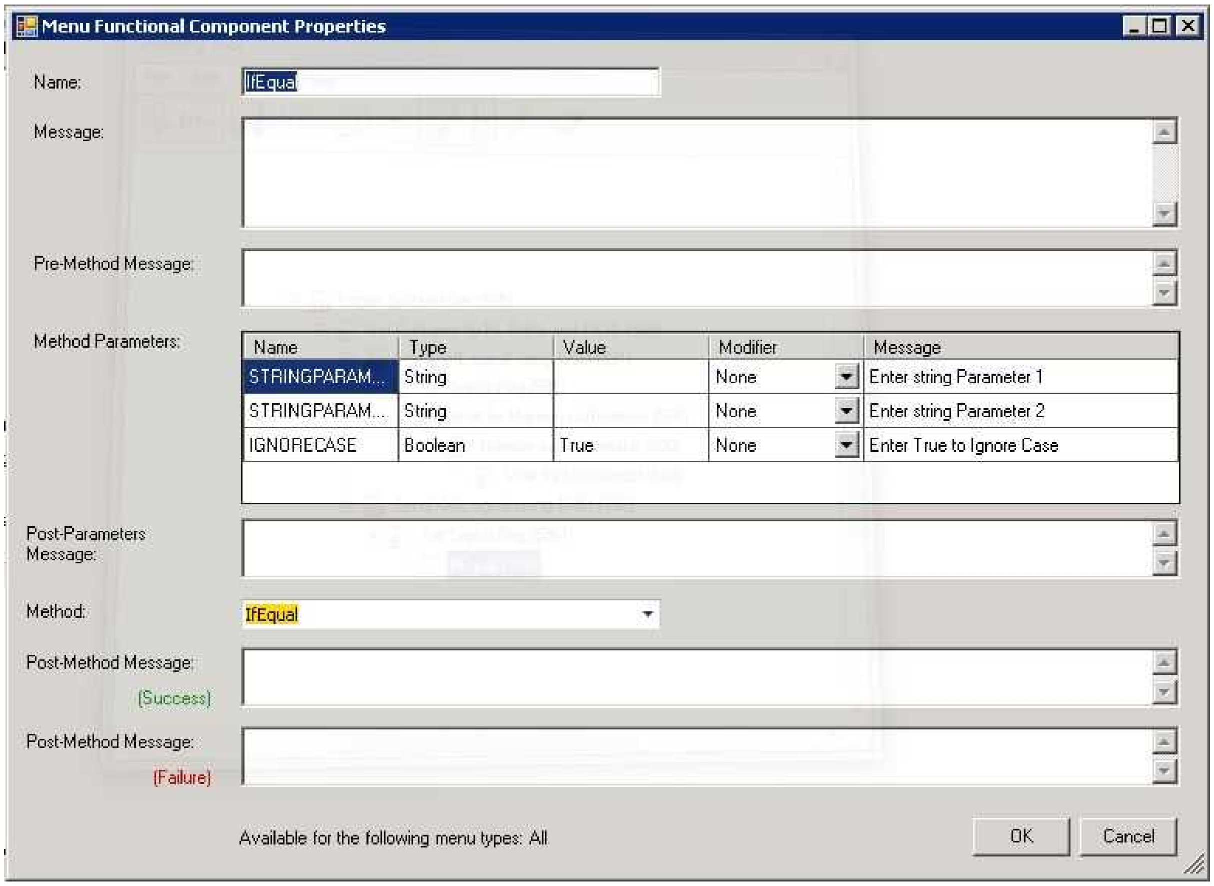
Comment:

5. Enter the necessary information into the fields on the "Menu Structure" form, and then click "**OK**". For more information about specific fields on the "Menu Structure" form, refer to the "Editing Menu Properties" document.

Result: the M.F.C. you added now displays on the Menu Structure Tree as shown in the following screenshot:


Logon System User (585)
Get Customer by RL Prefix and TX ID (588)
Send XML signal using TXID (591)
Set Signal Flag (595)
Wait for Manitou notifications (598)
Transfer call to operator (600)
Wait for Disconnect (602)
Send XML signal using TXID (592)
Set Signal Flag (596)
IfEqual (946)

6. Right-click the M.F.C. you added, and then select "Component Properties".
Result: the "Menu Functional Component Properties" form displays as shown in the following screenshot:


Menu Functional Component Properties

Name: IfEqual

Message:

Pre-Method Message:

Method Parameters:

Name	Type	Value	Modifier	Message
STRINGPARAM...	String		None	Enter string Parameter 1
STRINGPARAM...	String		None	Enter string Parameter 2
IGNORECASE	Boolean	True	None	Enter True to Ignore Case

Post-Parameters Message:

Method: IfEqual

Post-Method Message: (Success)

Post-Method Message: (Failure)

Available for the following menu types: All

7. Enter the necessary information into the fields on the "Menu Functional Component Properties" form, and then click "**OK**". For more information about specific fields on the "Menu Functional Component Properties" form, refer to the "Component Properties" document.

Standard M.F.C.s

Standard M.F.C.s can be divided into the following categories:

General

Dealer

Customer

Alarm

Telecom

Security

General

General M.F.C.s include any generic nodes that are regularly used as part of custom menu structures.

ClearCache

As the user moves through a menu, the data cache collects pieces of information (for e.g., the phone number from which the user is calling). Applying this component clears the data cache of this type of information.

ClearInputBuffer

The telephony input buffer is cleared of any key presses.

ClearOutputMessageBuffer

Any message previously sent to the system for speech or output (for e.g., SMS,TCP) that has not been delivered, is cleared.

EndSession

This component ends the session when other component requirements are met, or have failed. For example, a user might fail on three consecutive attempts to enter his user ID and password causing the session to end. Likewise, when a technician calls in to place an account On Test, the session will end after he completes all the necessary steps. The EndSession component has one parameter. Use this parameter to play a message before the session ends. By default, this parameter is set to be ignored. If you want to activate the parameter, enter the message you want and select "None" from the "Modifier" dropdown menu.

Exit
The Exit component is used to exit out of a menu. For example, the user may be prompted to exit a detailed menu and return to a main menu.

GetBooleanParamInput
This component gets an input of true or false, and stores the value in {$TBOOL1#}.

GetKeyPressed
This component returns when a key is pressed, or after the timeout period runs.

GetMultiParamInput
This component gets five inputs of strings and Boolean values. Strings are stored in {$TSTR1#}, {$TSTR5#} and Booleans are stored in {$TBOOL1#}, {$TBOOL5#}.

GetRow
This component gets a row from a selection. The GetRow function can be used to walk through multiple records and provide information to the user.

GetStringParamInput
This component gets an input of a string value and stores it in {$TSTR1#}.

HasKeyBeenPressed
This component returns true if a key has been pressed.

IfCounterEqual
This component tests to see if the value held in the counter is equal to a specific test value.

IfCounterGreater
This component tests to see if the value held in the counter is greater than a specific test value.

IfCounterGreaterEqual
This component tests to see if the value held in the counter is greater than or equal to a

specific test value.

IfCounterLess
This component tests to see if the value held in the counter is less than a specific test value.

IfCounterLessEqual
This component tests to see if the value held in the counter is less than or equal to a specific test value.

IfStringContains
This component tests to see if string 1 contains string 2.

IfStringEmpty
This component tests to see if string 1 is empty.

IfStringEndsWith*
This component tests to see if string 1 ends with string 2.

IfStringInList
This component tests to see if string 1 is in comma separated string 2.

IfStringStartsWith
This component tests to see if string 1 starts with string 2.

IfTrue
This component tests to see if a Boolean value is true.

IncrementCounter
This component increments the value held in the counter by one.

IsFirstRow
This component tests to see if this is the first row of the rows selected.

IsLastRow
This component tests to see if this is the last row of the rows selected.

IsTCPConnected
This component tests to see if a TCP connection is active.

None
This component is the default method when no processing is required.

RestoreSessionState

This component restores a session state to the previous input mode within a group (used with RegisterCallerIdWithSession). This allows a menu to go down a path and allows a user to DeregisterCallerIdWithSession, and then return the session to the previous state for other responding contacts.

SetCounter

This component sets the contents of a counter to a specified value.

SetVariableToSpeak

This component sets the contents variable into the {$SPEAK#} variable. This can then be played, and will be spelled out. So if the value is “EVEN123”, once it is in the {$SPEAK#} variable, the system does not attempt to pronounce the word, but speaks it one letter at a time (i.e., “E”,“V”,”E”,”N”,”1”,”2”,”3”).

Sleep

This component pauses execution for a specified time in milliseconds. One thousand milliseconds equals one second.

TCPConnect

This component opens a TCP connection to a server and port.

TCPDisonnect

This component closes a TCP connection.

TCPRead

This component reads the data sent on the TCP connection into {$TCPBUFFER#}.

TCPWrite

This component writes the data specified out of the TCP connection.

TTSConfiguration

This component tells the Telephony server to use a different configuration for text-to-speech. Users are prompted to enter the TTS Configuration number. By default, the system uses configuration "0". Each configuration controls how the voices are played and the voice that is used. By default, the system is configured for the Loquendo voice "Allison". This voice supports standard SAPI XML enhancements. Therefore, it has a header of "<SAPI>" and a footer of "</SAPI>". This is required so that the voice takes notice of the XML, and knows not just to read the text back.

ValueComparison

This component compares one value with another based on the comparison flag where:
= equals
< Less than
> Greater than
<= Less than or equal to
>= Greater than or equal to
If the comparison flag is "<", then the expression being evaluated is: "If value1 < value2 then True else False".

YesNo

This component returns either true or false depending on which key is pressed.

Dealer

Dealer M.F.C.s pertain specifically to Dealers as shown in the examples below.

AddActivityLogComment

This component adds a comment log line into the Manitou Activity Log.

ClearDealerCache

This component clears the Dealer data cache. It clears all Dealer records that have been previously read into the system.

GetDealerByContID

This component gets the Dealer record using the Dealer's ID.

GetDealerbyPhoneNumber

This component gets the Dealer record using the Dealer's phone number.

LoadDealerByDNIS

This component gets the Dealer record from the Dealer's caller ID.

Customer

Customer M.F.C.s include components for retrieving Customer information and putting an account On Test.

ClearCustomerCache

As the Customer moves through the menu, the data cache collects pieces of information (for e.g., the phone number from which the Customer is calling). Using this functional component clears the data cache of this type of information.

CommissionCustomer

This component changes the commission state of the Customer.

GetCustomerArea

This component gets a Customer area record.

GetCustomerByContID

This component prompts the user to enter a contract ID for identification.

GetCustomerByPhoneNumber

This component prompts the user to enter a phone number for identification.

GetCustomerByRLPrefixTxID

This component prompts the user to enter his Receiver Line Prefix and the Transmitter ID. Then, it retrieves Customer account information.

GetCustomerBySerialNo

This component prompts the user to enter a serial number. This is a unique ID by which all Manitou Customers are referenced.

GetCustomerContactByPhoneNumber

This component prompts the user to enter a contact phone number. The component can search at the Customer level or in contacts. You can also specify a serial number.

GetCustomerForSelectedTicket

This component prompts the user to enter a serial number. This is a unique ID by which all Manitou Customers are referenced.

GetCustomerOnTestParameters

This component checks to see if there is a match to the parameters entered in the Customer's On Test details.

GetCustomerOnTestStatus

This component gets the Customer's On Test status.

GetCustomerZone
This component gets the Customer zone record.

GetLastCustomer
This component gets the last Customer accessed by the user in the previous eight hours.

GetSingleCustomerAccess
If the user logged into the system has access to a specific single Customer, this component returns that Customer.

GetTicket
This component gets a specific ticket for the integrated service application.

MarkTicketListenedTo
This component marks On Test signals after they have been listened to. Use with SelectOnTestSignals. You must select the "Keep signals for OpenVoice access" checkbox.

MarkSignalXmittedToTech
This component marks On Test signals after they have been transmitted to a technician. Use with SelectOnTestSignals. You must select the "Keep signals for OpenVoice access" checkbox.

PutCustomerFullyOnTest
This component puts the whole Customer On Test.

PutCustomerPartiallyOnTest
This component puts part of a Customer system On Test.

RegisterForSMSNotify
This component registers a user to be notified by SMS of signals that arrive while the account is On Test.

SelectCustomerArea
This component selects a single Customer area. Use the GetRow component to manipulate the results.

SelectCustomerByGroupCode
This component selects Customers by group code. Use the GetRow component to manipulate the results.

SelectCustomerByPhoneNumber

This component selects Customers by phone number. The system looks for phone numbers on the Customer details page first. Then, if it does not find any, it looks for contacts attached to the Customer record for a match. Use the GetRow component to manipulate the results.

SelectCustomerContactList

This component selects contacts "CpType". These are user-definable in Manitou. By default, however, they are Mobile, Phone, Home, Business, etc. Include other contacts on the contact list by setting the flag (for e.g., if you require Branch contacts, set "IncBranch" to true). Use the GetRow component to manipulate the results.

SelectCustomerLog

This component selects Customer Activity Log signals and alarm rows for a specific number of minutes in the past - first by system and Transmitter numbers, then by Event or Event Category. Use the GetRow component to manipulate the results.

SelectCustomerOnTestDetailRows

This component selects a single Customer's On Test detail rows. Use the GetRow component to manipulate the results.

SelectCustomerZones

This component selects a single Customer's zones. Use the GetRow component to manipulate the results.

SelectOnTestSignals

This component selects a Customer's On Test signals. Use the GetRow component to manipulate the results.

SelectTicketList

This component selects a technician's ticket list from the integrated service application. Use the GetRow component to manipulate the results.

SelectTransmitterList

This component selects a single Customer's list of transmitters. Use the GetRow component to manipulate the results.

StoreLastCustomer

This component stores the last Customer for a user so the system can retrieve it with the GetLastCustomer call.

TakeOffTest

This component takes the Customer completely Off Test.

UnregisterForSMSNotify

This component unregisters a user who was previously registered with RegisterForSMSNotify. This results in the user no longer receiving SMS messages for signals that come in On Test.

Alarm

Alarm M.F.C.s enable you to perform various alarm-related functions (for e.g., sending signals, searching for Customers, and entering/exiting different alarm modes).

AddAlarmComment

This component adds a comment line into the Manitou Activity Log.

CancelAllAllocatedAlarms

This component closes all alarms allocated to a user. The user must be in alarm handling mode for this to be successful.

CancelCloseAlarms

This component closes the current alarm allocated to the user. The user must be in alarm handling mode for this to be successful.

DidCustomerDisconnect

This component returns true if the Customer disconnected.

EnterAlarmHandling

This component puts the user into alarm handling mode.

ExitAlarmHandling

This component removes the user from alarm handling mode.

ExtendCloseTime

This component extends the close time for an area by a number of minutes.

ExtendOpenTime

This component extends the open time for an area by a number of minutes.

GetAlarm

This component gets a specified alarm by its alarm number.

GetCustomerForAlarm

This component gets a single Customer for the current alarm. This is used primarily by outbound OpenVoice and outbound AutoText. It is initiated from a Manitou Action Pattern.

PreCancelAlarm

This component pre-cancels an alarm through the system. This functions the same as the Manitou pre-cancel command.

RecordVoiceMessageAlarm

This component records a voice message for an alarm.

SelectAlarmsByCallerID

This component selects and allocates alarms by caller ID.

SendInvalidMessage

This component sends an invalid message (%X) into Transmitter ID "0", zone "1".

SendReceiverContactID

This component sends and receives a contact ID signal using DTMF tones.

SendSignal

This component sends a generic two-way signal into Manitou. It can be flagged as a two-way signal. This is used in the InstantConnect functionality.

SendSignalWithAudio

This component records a message for a caller currently online. Then, the system sends it into Manitou as a signal.

Telecom

Telecom M.F.C.s have been provided to assist users in telephony endeavors (for e.g., setting modes and disconnecting).

AddCallToACDQueue

This component adds an incoming call into an ACD queue.

BusyPort
This component makes the line port ring as busy.

FlushSMSBuffer
This component flushes any text waiting to be sent to an SMS device.

Hangup
This component hangs up the call.

IsCallInProgress
This component tests to see whether a call is in progress. The component returns true if the call is connected to an Operator, or if the call is on hold. It returns false if the Customer is disconnected.

OpenVoiceRegisterCallSession
This component registers an OpenVoice call with Manitou.

PlayTones
This component sends a DTMF tone.

PlayWaveFile
This component plays a predetermined .wav file.

SetCallResult
This component sets the result of a call to success or failure. The value is stored in {$RSUC#}, and the comment is stored in {$RERRS#}.

SetListenMode
This component plays tones that put the system into listen mode.

SetTalkMode
This component plays tones that put the system into talk mode.

TransferCall
This component transfers a call to a specified number.

TwoWayTransferCall
This component transfers a two-way call to a specified number.

WaitForDisconnectFromTransfer

This component waits for a call to disconnect from a transfer.

Security

Security M.F.C.s pertain to system and account accessibility.

CanUserCancelAlarm

This component determines whether the user has the permissions necessary to cancel an alarm.

CanUserPutOnTest

This component determines whether the user has the permissions necessary to put the system On Test.

DoesUserHaveCustomerAccess

This component determines whether the user has the permissions necessary to access the Customer. The Customer must have already been loaded for this to be successful.

IsRegisteredCallerID

This component determines whether a caller ID is registered in the MediaGateway 2. The system returns true if the caller ID is registered.

LogOn

This component logs the user onto Manitou after verifying his credentials.

RegisterNewUser

This component registers security details for a new user. It allows users to register with the extra security of caller ID without increasing data entry for the Central Station.

SystemLogOn

This component logs the user onto Manitou as a MediaGateway 2 user. There is no security validation. This component should not be used for user system access. Use the LogOn component so the system can validate the user's credentials.

TwoWayPLUS M.F.C.s

The TwoWayPLUS M.F.C.s pertain to two-way call options.

AgentRegisterCallSession
This component registers a two-way call with Manitou. The call then displays in the Manitou Call Queue.

SoloProtectRegisterCallSession
This component registers a SoloProtect call with Manitou.

TwoWayAdoptCallSession
This component adopts an existing two-way call session into Manitou. This is used for two-way direct when the call hits the Receiver and then flash hooks to the MediaGateway 2. The call needs adopting because the FEP (and not the MediaGateway 2) creates the session.

TwoWayRegisterCallSession
This component registers a two-way call with Manitou.

TwoWayUnregisterCurrentSession
This component unregisters a two-way call with Manitou.

TwoWayWaitForDisconnectFromTransfer
This component waits for a two-way call to disconnect from a transfer.

TwoWayWaitForManitouNotifications
This component waits for notifications from Manitou about the call (for e.g., the extension to which the call should be transferred or the state of the call, if it has dropped). The component returns true if a transfer is requested. It returns false if the call times out. It throws an exception and terminates if the call disconnects.

TwoWayWaitForReceiver
This component waits for the Receiver to pass the two-way call into the MediaGateway 2.

WaitForReceiver
This component waits for the Receiver to pass the two-way call into the MediaGateway 2.

SoloProtect M.F.C.s

SendAudioSliceSignal
This component sends a sliced audio signal in the slice duration you set.

WaitForAudioSlice
This component waits for a sliced audio signal to be received.

AutoText/UC M.F.C.s

DeregisterCallerIDWithSession
This component deregisters a phone number from a group. Once deregistered, the caller ID no longer receives text messages.

RegisterCallerIDWithSession
This component registers a phone number to a group. Then, the group can send and receive SMS messages as a group.

SendMappedSignal
This component identifies the mapping to apply to a signal that will be processed.

SendSMS
This component sends a text message.

UCDeleteNextSignal
This component deletes the next signal and prevents it from being sent into Manitou.

UCIsGPSsignal
This component determines whether the signal is a GPS signal.

UCIsNextSignal
This component determines whether there is a next signal.

UCModifyEvent
This component modifies the Event Code after a signal has been prepared with UCPrepareSignal.

UCModifyZone
This component modifies the zone after a signal has been prepared with UCPrepareSignal.

UCPrepareSignal
This component identifies the mapping to apply to a signal that was received and will be processed. This component prepares the signal to be sent, but does not send it. You can still use UCModifyEvent and UCModifyZone before you send the signal using UCSendPreparedSignal.

UCSendPreparedSignal
This component sends a signal prepared with UCPrepareSignal.

UCSendSignal
This component identifies the mapping to apply to a signal that was received and will be processed. This call prepares the signal to be sent, and then sends it.

SocialMedia M.F.C.s

SendSignalSocialMedia
This component sends a Social Media signal into Manitou.

UCInGeoFence
This component determines whether a signal originates from inside the GeoFence.

UniversalConnector

Bold’s UniversalConnector receives signals from sources such as SMS, GPRS, email, ODBC databases, FTP, RSS, TCP, UDP and simple files. It then transforms them into conventional signals (i.e., Event, GPS, video, and audio alarms), and delivers them into Manitou. Essentially, the UniversalConnector acts as a Receiver for non-traditional transmissions.

The following section illustrates how to set up and use the UniversalConnector with the MediaGateway 2 to receive alarms generated through non-traditional sources into the Manitou system. Additionally, the UniversalConnector provides a mechanism to communicate outwardly with Alarm Panels that allow SMS communication. This enables features such as Arm and Disarm.

UniversalConnector Requirements

The following items are required for the UniversalConnector:

- MediaGateway 2
- Licenses for each connector module

Data Mapping

Data Mapping consists of taking a signal that has arrived through a connector, breaking it apart according to a defined method, and then reassigning the fields to those associated with a Manitou alarm signal. A mapping template is independent of a connector (for e.g., the same template can be used for both email and SMS text signals).

Mapping Types

The first decision you need to make when creating a data map for a signal is what mapping type you want to use. The following four mapping types are available:

1. **Separator -** is a signal that uses a single character to separate the fields in a signal (for e.g., if a signal is received that contains the Transmitter ID, the Event Code, the area, and the zone all separated by a comma, the comma is acting as the separator). The signal is represented by the following: "12345,BA,1,9".

Note: the ODBC type connector delivers its data in a comma separated format. The connector performs a “select * from table". It concatenates the data returned together with a comma separator.

2. **Position -** this signal uses position and length to divide up a signal. An example of a Position Type signal is represented by the following: "12345BA19". To break it apart, you must know the following information: the Transmitter ID starts at character 1 and is 5 characters long, the Event Code starts at character 6 and is 2 characters long, the area starts at character 8 and is 1 character in length, and the zone starts at character 9 and is one character in length. For this method to work, all signals have to be formatted in the same manner.

3. **Label/Separator** – this signal has a label and then a data value. A separator (usually a carriage return) partitions the first value from the next label. The following is an example of a Label/Separator type:

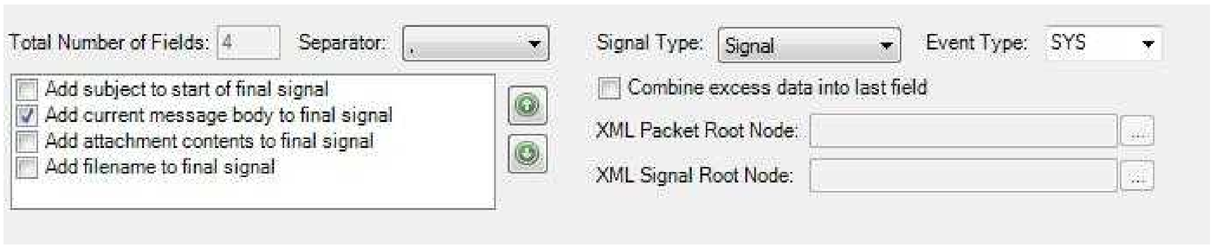
"Transmitter ID: 12345
Event: BA
Area: 1
Zone: 9".

4. **XML -** this type uses the standard XML (Extensible Markup Language) format. The XML type is represented by the following:

```
<?xml version="1.0"?>
<Alarms>
   <Customer TxId="12345" >
      <Signal Event="BA">
         <Area>1</Area>
 <Zone>9</Zone>
      </Signal>
   </Customer>
  </Alarms>
```

This format is very flexible so the signal could be represented in other ways, for example:

```
<?xml version="1.0"?>
<Alarms TxId="12345">
          <Signal Event="BA" Area = "1" Zone="9" />
</Alarms>
```


Total Number of Fields: 4
Separator: ,
Signal Type: Signal
Event Type: SYS
[ ] Add subject to start of final signal
[✓] Add current message body to final signal
[ ] Add attachment contents to final signal
[ ] Add filename to final signal
[ ] Combine excess data into last field
XML Packet Root Node:
XML Signal Root Node:

XML Type

Other formatting options generally depend on which mapping type you want to use. The separator is only relevant for Separator and Label/Separator data map types. The signal type can be a Signal, GPS Signal, or Telemetry. This field controls the fields to which you can map a signal. A regular signal does not allow latitude, longitude, speed, heading, or any other GPS type information. You must have a license to create a GPS signal. Using telemetry is a way to distinguish the signal from alarm type conditions, and to show that it is an informational signal. This is not currently differentiated in Manitou.

The Event Type specifies to which Manitou Event Map the signal belongs. Manitou uses the following three Event Types:

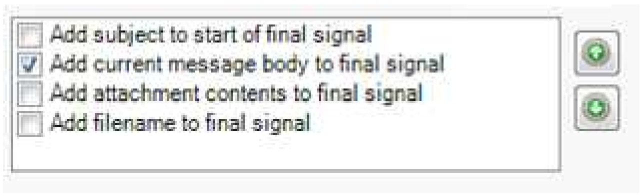
- SYS: Manitou standard system codes
- SIA: SIA standard codes
- CID: Contact ID standard codes

You can, however, enter your own code. In order for the code to be correctly decoded, you must enter it into Manitou.

Signal Parts

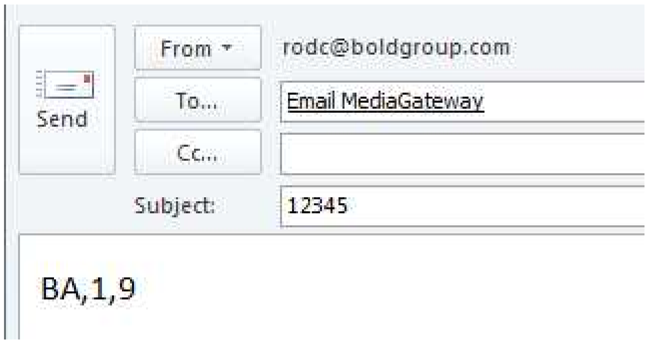
Depending on the connector, a signal can have multiple parts. For example, an email may have a body, subject, and attachments. Each of these pieces of information contains data that can be decoded in the data mapping process.

The signal parts box specifies what should be processed, and in what order. So, if an email that is using the Separator mapping type is sent into a data mapping template, you must define how to put the final signal together as shown in the following screenshot:



- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

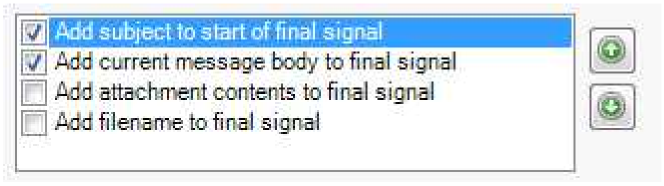
Note: you must select at least one of the checkboxes shown above.



From: rodc@boldgroup.com
To: Email MediaGateway
Subject: 12345

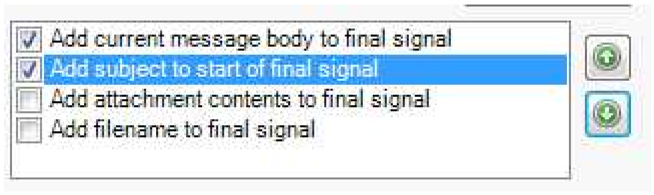
BA,1,9

In the email above, the Transmitter ID displays in the subject, and the event, area, and zone display in the body. To process the signal, it is necessary to put it all together in the correct order. So, we check the subject and body options, and then use the green arrows to move the lines up or down into the correct order. The order is important because it defines how Manitou processes the signal.



- [x] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

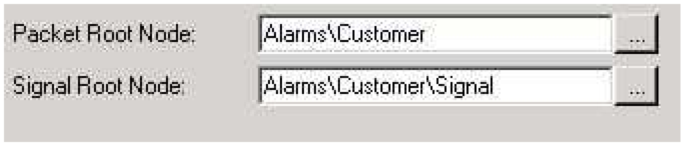
This creates the following signal: - **12345,BA,1,9**


- [✓] Add current message body to final signal
- [✓] Add subject to start of final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

This creates the following signal: - **BA,1,9,12345**

- [ ] Combine excess data into last field

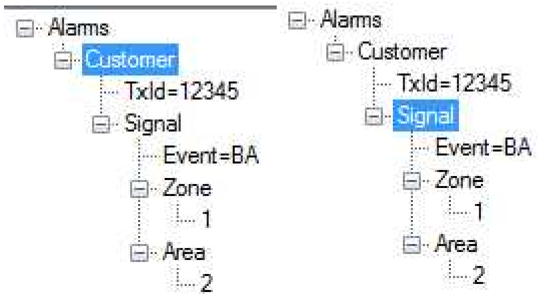
Use the "Combine excess data into last field" option to lump all remaining data into the last field. This can be useful if the system is sending erroneous information (for e.g., signatures or unwanted data) at the end of the signal. This can be collected or ignored in the last field.


Packet Root Node: Alarms\Customer

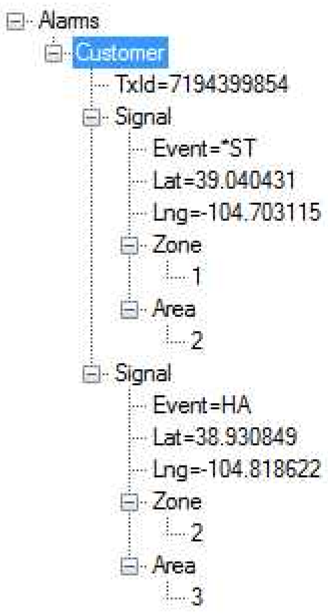
Signal Root Node: Alarms\Customer\Signal

If you are using the XML mapping type, the "Packet Root Node:" and "Signal Root Node:" fields enable you to specify the parts of the signal required for processing. The packet node identifies the start of the packet of information. Within this packet, there might be multiple signals. The signal root node identifies where those signals are.

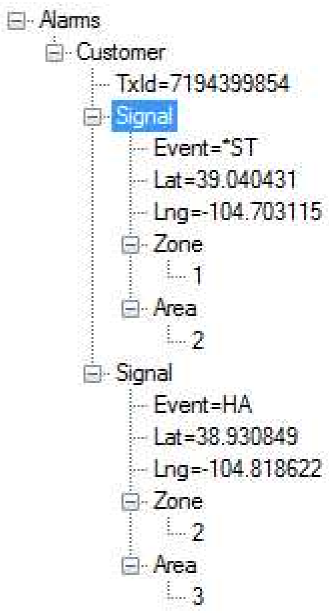
In our simple example, these items are identified below. Nodes are highlighted.


```
Alarms                          Alarms
  Customer                        Customer
    TxId=12345                      TxId=12345
    Signal                          Signal
      Event=BA                        Event=BA
      Zone                            Zone
        1                               1
      Area                            Area
        2                               2
```

A more complex GPS signal with multiple parts is shown below. Nodes are highlighted.



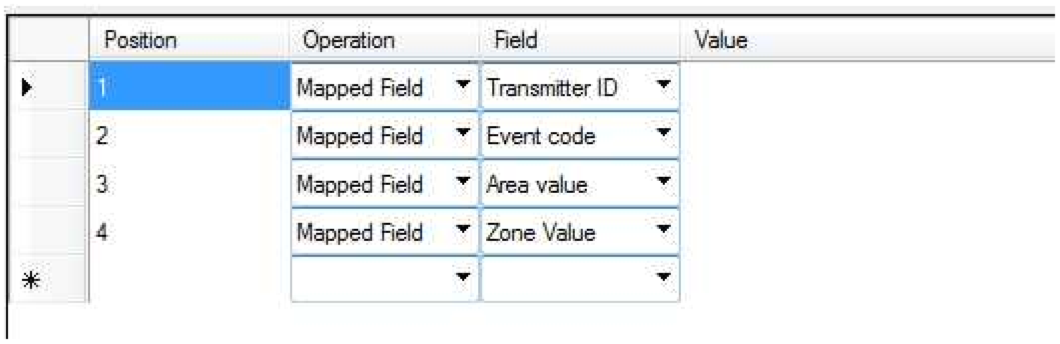
```
Alarms
  Customer
    TxId=7194399854
    Signal
      Event=*ST
      Lat=39.040431
      Lng=-104.703115
      Zone
        1
      Area
        2
    Signal
      Event=HA
      Lat=38.930849
      Lng=-104.818622
      Zone
        2
      Area
        3
```



```
Alarms
  Customer
    TxId=7194399854
    Signal
      Event=*ST
      Lat=39.040431
      Lng=-104.703115
      Zone
        1
      Area
        2
    Signal
      Event=HA
      Lat=38.930849
      Lng=-104.818622
      Zone
        2
      Area
        3
```

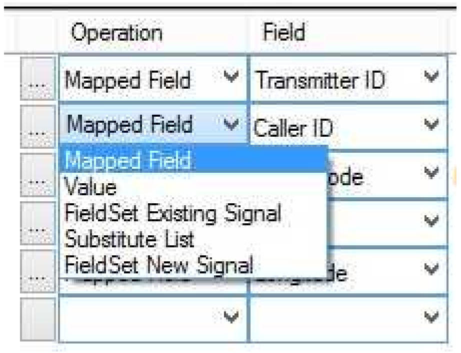
Mapping Fields

Field mapping varies depending on the mapping type being used. In the example below, we are mapping the comma separated signal identified above.

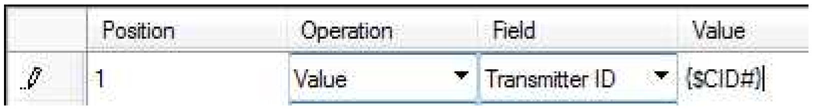


	Position	Operation	Field	Value
▶	1	Mapped Field	Transmitter ID	
	2	Mapped Field	Event code	
	3	Mapped Field	Area value	
	4	Mapped Field	Zone Value	
*				

The position field is numeric and specifies the order of the fields. The operation has the following five options:

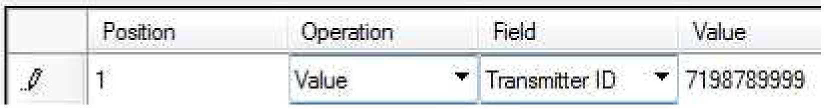


Mapped: select this option to accept the data contained in the signal.
Value: select this option to enter a value for the data or to use a system variable. The value is then entered in the "Value" column. For example, if the device sending the signal is a cellular device and you want to use the caller ID of the device as the Transmitter ID, you can do this using “Value Operation”.



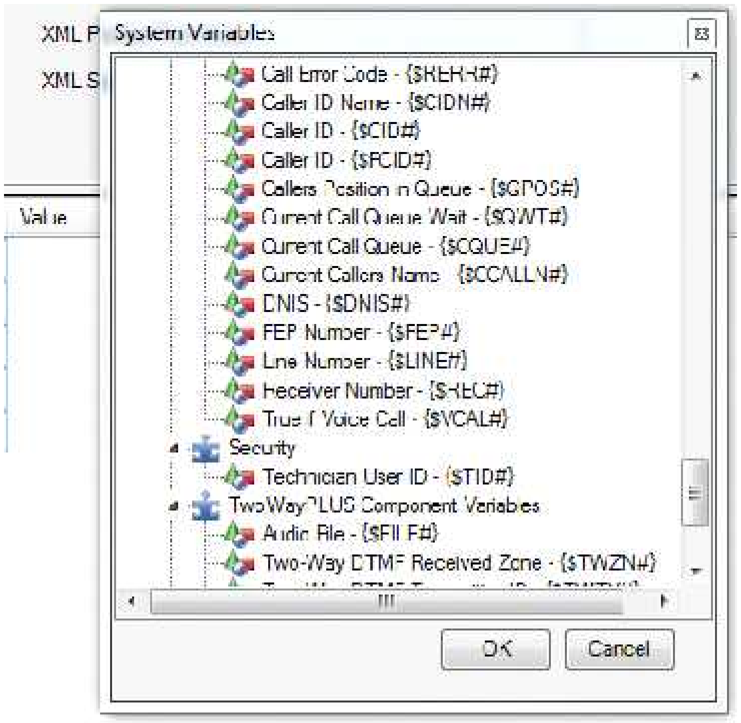
	Position	Operation	Field	Value
	1	Value	Transmitter ID	{$CID#}

You can also hard code a value for testing. See the example screenshot below:

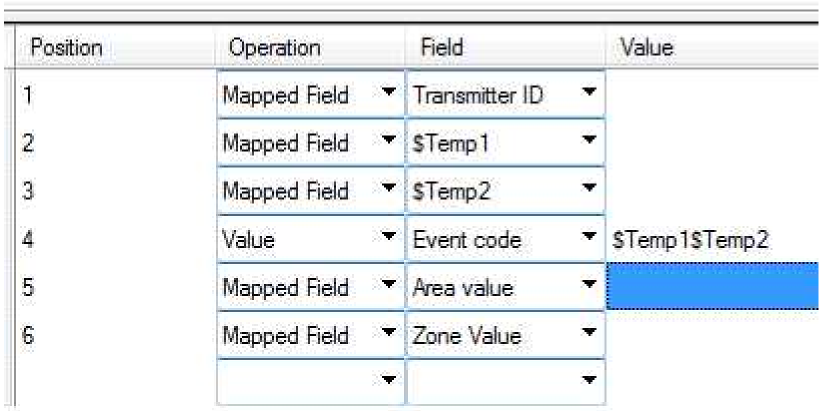


	Position	Operation	Field	Value
	1	Value	Transmitter ID	7198789999

{$CID#} is the caller ID for the device. You do not have to remember these variables. Left-click the values field so that the cursor displays as flashing. Then, right-click to display the “System Variables” window. Select the variable you want.


XML P
XML S
Value
System Variables
Call Error Code - {$RERR#}
Caller ID Name - {$CIDN#}
Caller ID - {$CID#}
Caller ID - {$FCID#}
Callers Position in Queue - {$GPOS#}
Current Call Queue Wait - {$QWT#}
Current Call Queue - {$CQUE#}
Current Callers Name - {$CCALLN#}
DNIS - {$DNIS#}
FEP Number - {$FEP#}
Line Number - {$LINE#}
Receiver Number - {$REC#}
True if Voice Call - {$VCAL#}
Security
Technician User ID - {$TID#}
TwoWayPLUS Component Variables
Audio File - {$FILE#}
Two-Way DTMF Received Zone - {$TWZN#}
OK
Cancel

The value is also useful for manipulating fields. If you have a signal that sent an Event Code in two parts, and you wanted to combine them into one. You could do it as follows:
Signal: 12345,F,A,1,9

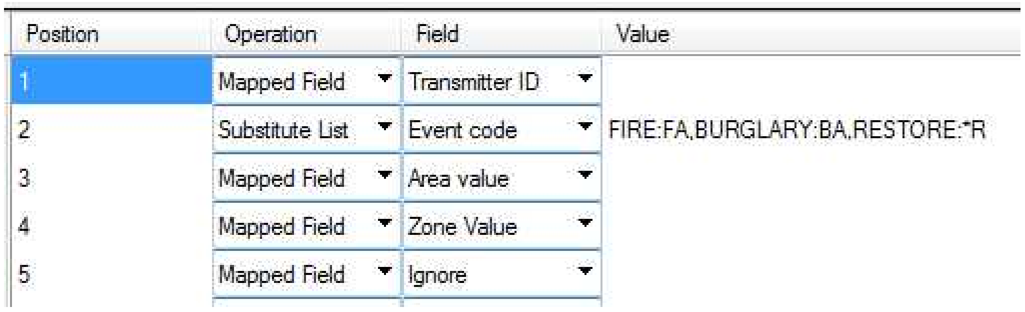


Position	Operation	Field	Value
1	Mapped Field	Transmitter ID	
2	Mapped Field	$Temp1	
3	Mapped Field	$Temp2	
4	Value	Event code	$Temp1$Temp2
5	Mapped Field	Area value	
6	Mapped Field	Zone Value	

Here, we are mapping the two parts of the Event Code into a temporary variable. Then using the value, we assign them to the Event Code. By placing them straight after each other in the value field, they are concatenated together. So, “F” and “A” becomes “FA”.

Substitute List: select this option to change the characters necessary to define the values. Take our current example, but assume that the Event Code was more verbose. It might send Event Codes of FIRE, BURGLARY, RESTORE etc. The Substitute List allows us to modify the Events into a shorter form.

So, we might want FIRE à FA, BURGLARY à BA, RESTORE à *R.



Position	Operation	Field	Value
1	Mapped Field	Transmitter ID	
2	Substitute List	Event code	FIRE:FA,BURGLARY:BA,RESTORE:*R
3	Mapped Field	Area value	
4	Mapped Field	Zone Value	
5	Mapped Field	Ignore	

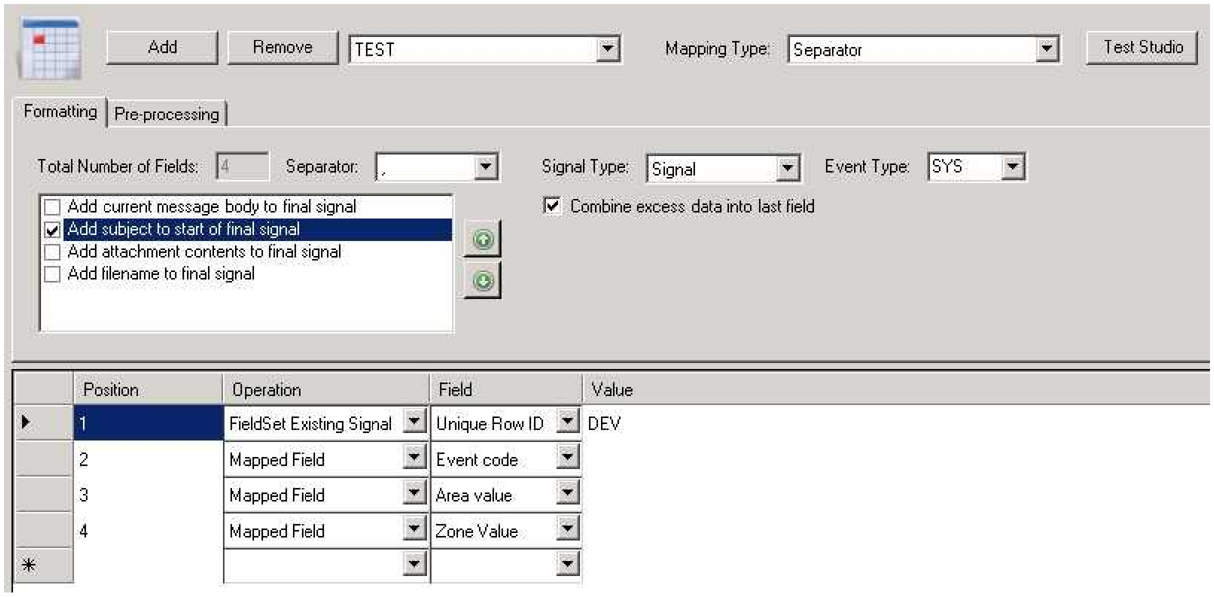
To achieve this, we create a Substitute List as shown above. We are using a comma to separate the items, and a colon to link the substitutions.

FieldSet: the FieldSet option allows you to set up data mapping templates that reference other templates. This can be useful when you have a device that sends signals in multiple formats. This requires a different template for each format. One way to handle multiple formats is to send them into different modems at the receiving center. This, however, is not very scalable. Another way is to include an identifier in the format you are sending, as shown in the following example:

The following comma separated signals use the first field to determine the format. The signal with an S1 is a regular signal. The signal with an S2 is a GPS location signal.

S1, 12345, BA, 1, 9
S2, 12345, 38.941286, -104.719182

Using the FieldSet, we can create an overlay template that loads the appropriate template for the signal being processed. We need to create two templates to handle the signals above, and a third to handle the FieldSet mapping. We create the overlay template only to read the Signal Type field.



TEST
Mapping Type: Separator
Total Number of Fields: 4
Separator: ,
Signal Type: Signal
Event Type: SYS

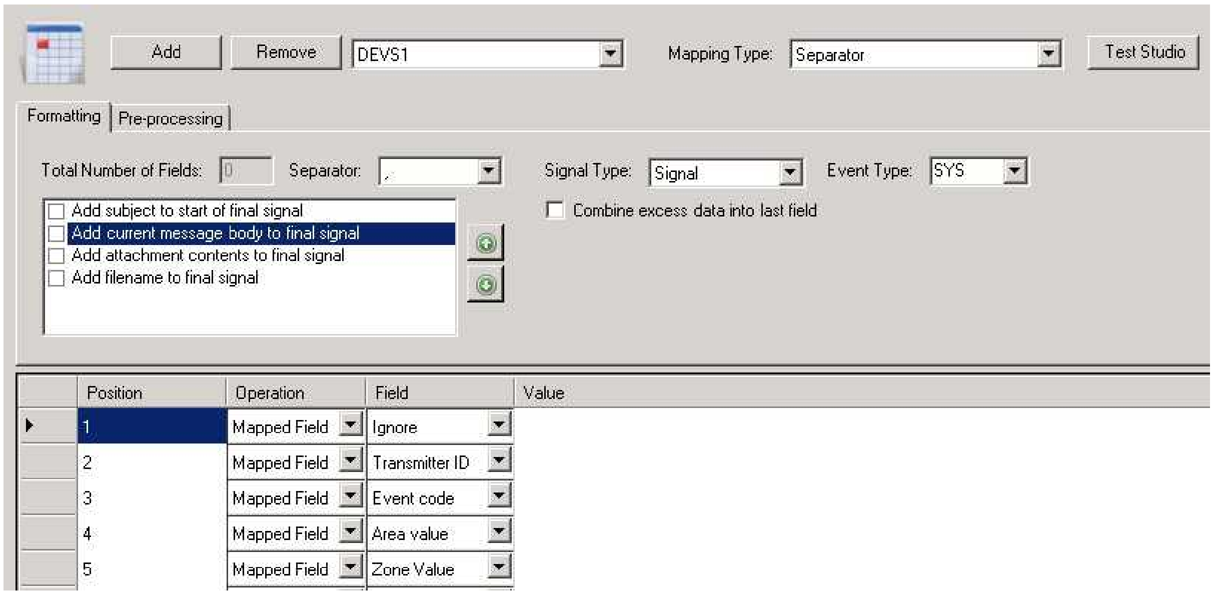
- [ ] Add current message body to final signal
- [x] Add subject to start of final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [x] Combine excess data into last field

Position	Operation	Field	Value
1	FieldSet Existing Signal	Unique Row ID	DEV
2	Mapped Field	Event code	
3	Mapped Field	Area value	
4	Mapped Field	Zone Value	

The operation is FieldSet. It attempts to load another template using the name found in the field (i.e., S1 or S2). This can be modified by entering a tag in the "Value" column. This prepends onto the value. In this example, the system looks for the template of DEVS1 or DEVS2.

The DEVS1 template is necessary to process the regular signal. It looks like this:



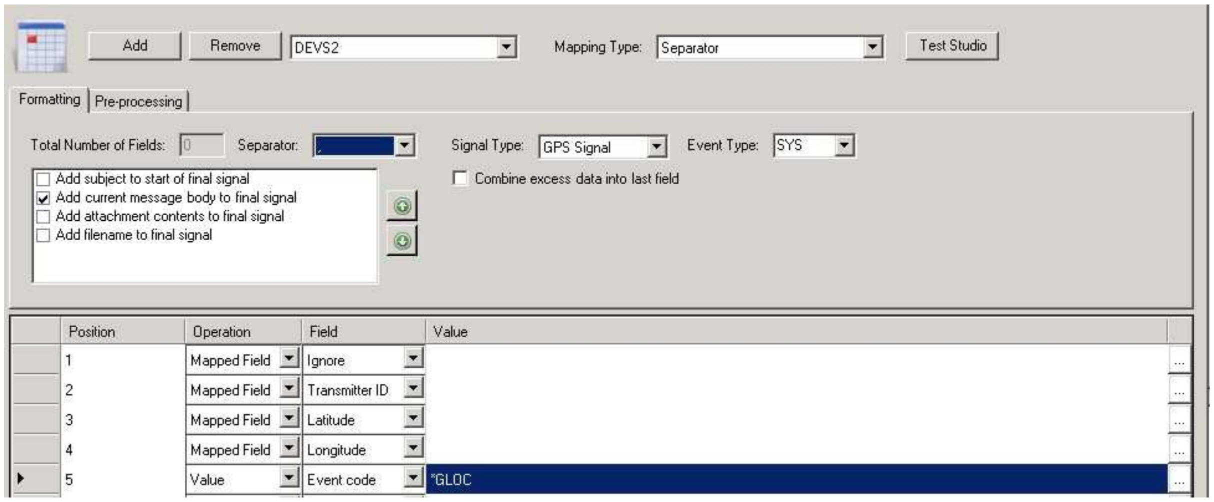
DEVS1
Mapping Type: Separator
Total Number of Fields: 0
Separator: ,
Signal Type: Signal
Event Type: SYS

- [ ] Add subject to start of final signal
- [ ] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [ ] Combine excess data into last field

Position	Operation	Field	Value
1	Mapped Field	Ignore	
2	Mapped Field	Transmitter ID	
3	Mapped Field	Event code	
4	Mapped Field	Area value	
5	Mapped Field	Zone Value	

The DEVS2 template is necessary to process the GPS signal. It looks like this:



Add | Remove | DEVS2 | Mapping Type: Separator | Test Studio

Formatting | Pre-processing

Total Number of Fields: 0 | Separator: | Signal Type: GPS Signal | Event Type: SYS

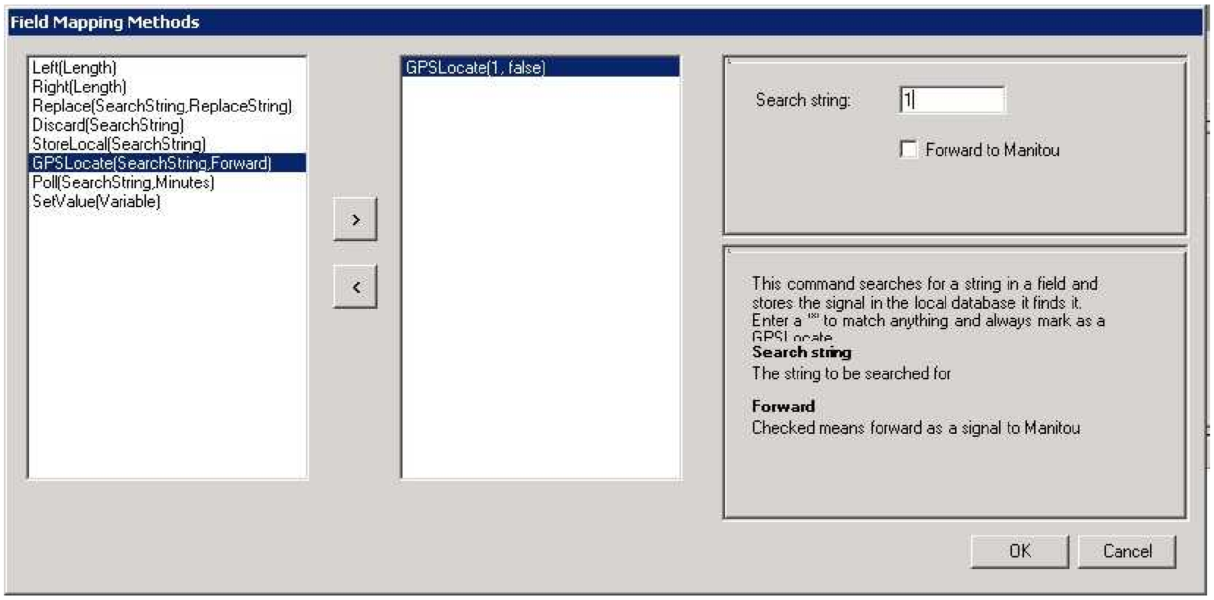
- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal
- [ ] Combine excess data into last field

Position	Operation	Field	Value
1	Mapped Field	Ignore	
2	Mapped Field	Transmitter ID	
3	Mapped Field	Latitude	
4	Mapped Field	Longitude	
5	Value	Event code	"GLOC

Notice in both signals that the first field is set to be ignored. This is the signal type used by the SMS template to find the appropriate template for the signal. However, it is not needed to process the signal.

Field Mapping Methods

Double-click in the "Value" cell to display the "Field Mapping Methods" dialog displayed in the following screenshot:



Field Mapping Methods

Left(Length)
Right(Length)
Replace(SearchString,ReplaceString)
Discard(SearchString)
StoreLocal(SearchString)
GPSLocate(SearchString,Forward)
Poll(SearchString,Minutes)
SetValue(Variable)

GPSLocate(1, false)

Search string: 1
- [ ] Forward to Manitou

This command searches for a string in a field and stores the signal in the local database it finds it. Enter a "" to match anything and always mark as a GPSLocate
Search string
The string to be searched for
Forward
Checked means forward as a signal to Manitou

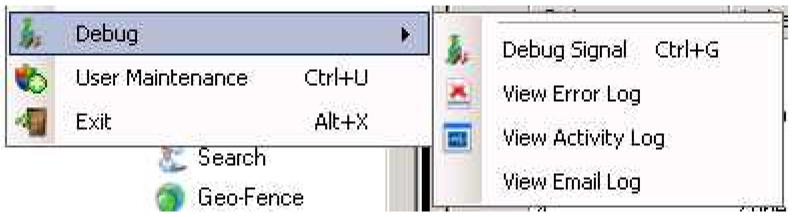
OK | Cancel

Left(Length) – this method takes the character length of a field and makes it the current value. So, if a field contains the value “FIRE” or “BURGLARY”, the Left(1) modifies the method to an “F” or “B”.
Right(Length) – this method works like the Left(Length) method, but takes from the right

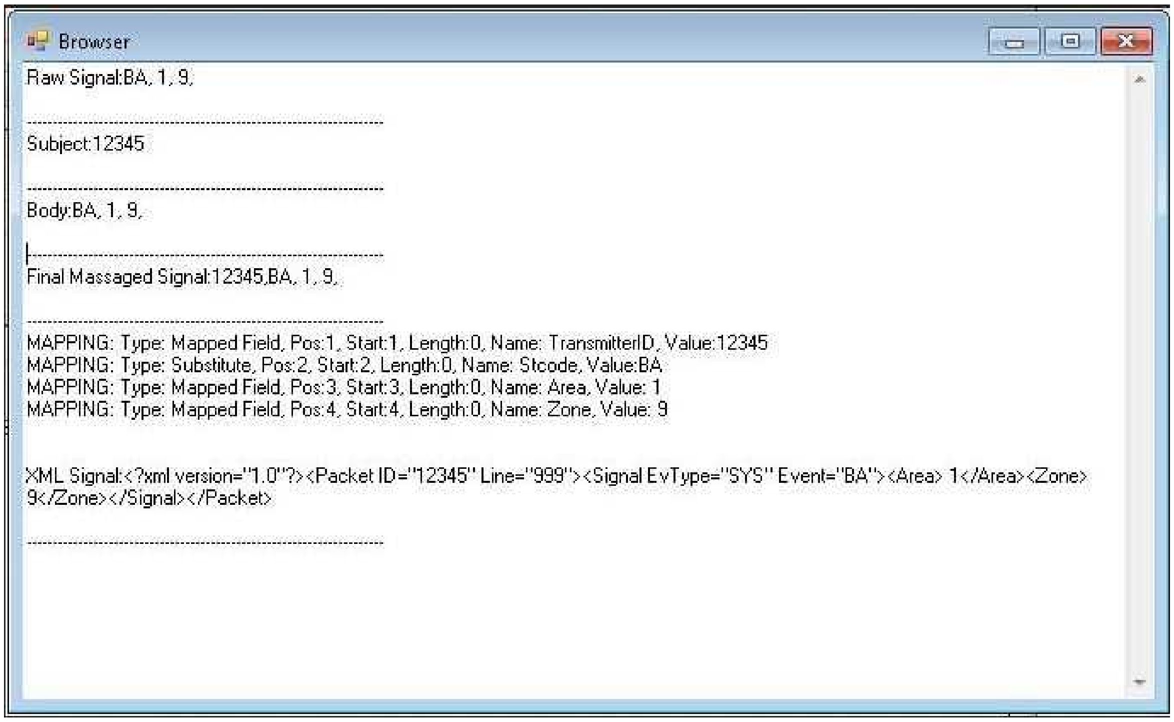
side of the field.
Replace(SearchString, ReplaceString) – this method replaces one string with another. "Replace(FIRE,FA)" replaces the string “FIRE” with “FA”. It reports partial matches, so “FIREALARM” does not get changed to “FAALARM”.
Discard(SearchString) – this method results in the signal being discarded if the field matches the search string.
Store(SearchString) – this method results in the signal being stored in the MG_SIGNALS table if the field matches the search string.
GPSLocate(SearchString, Forward) – this method results in the signal being stored in the MG_SIGNALS table if the field matches the search string as a GPSLocate signal. GPS signal processing (for e.g., checking against Geo-fences) occurs. If "Forward to Manitou" is checked, the signal forwards to Manitou.
Poll(SearchString, Minutes) - if the field matches the search string, this method results in the signal being counted as a polling signal for the device sending it. If the device does not send another polling signal within the number of minutes you define, an alarm triggers in Manitou for the device. A device must send at least one polling signal in order to register. If the device fails to poll, only one alarm will be sent to Manitou. The device will have to signal again to restart the process.

Debugging

Debugging is the mechanism used to trace what happens to a signal when it is processed. To send a Debug Signal into the UniversalConnector, press CTRL-G or choose the following option from a File menu:



This results in the browser window displaying the processing shown in the following screenshot. It displays the Raw Signal and the subject. Then, it shows how they are combined together. Next, it shows how each field is processed and the value for each field. Finally, it shows the XML signal sent to Manitou. These pieces of information enable you to determine how the signal is transformed as it goes through the UniversalConnector.



```
Raw Signal:BA, 1, 9,

-----------------------------------------------------------------------
Subject:12345

-----------------------------------------------------------------------
Body:BA, 1, 9,

-----------------------------------------------------------------------
Final Massaged Signal:12345,BA, 1, 9,

-----------------------------------------------------------------------
MAPPING: Type: Mapped Field, Pos:1, Start:1, Length:0, Name: TransmitterID, Value:12345
MAPPING: Type: Substitute, Pos:2, Start:2, Length:0, Name: Stcode, Value:BA
MAPPING: Type: Mapped Field, Pos:3, Start:3, Length:0, Name: Area, Value: 1
MAPPING: Type: Mapped Field, Pos:4, Start:4, Length:0, Name: Zone, Value: 9


XML Signal:<?xml version="1.0"?><Packet ID="12345" Line="999"><Signal EvType="SYS" Event="BA"><Area> 1</Area><Zone>
9</Zone></Signal></Packet>

-----------------------------------------------------------------------
```

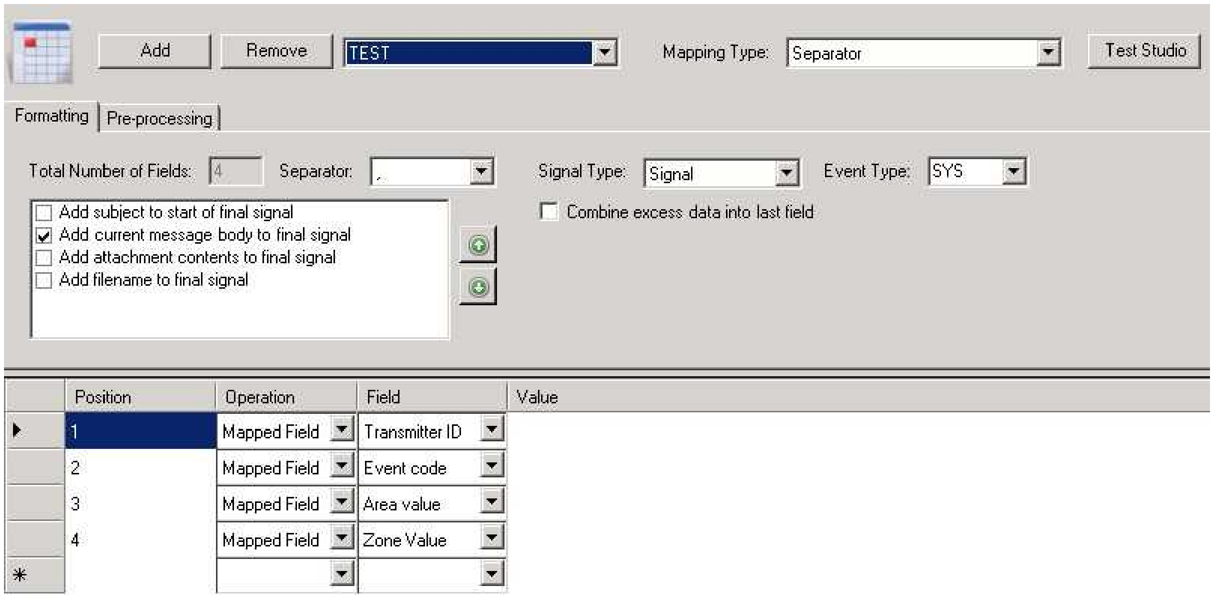
Signal Debugging

Data Mapping Examples

The following are example data maps for the mapping types discussed above.

Separator

The Separator signal for the template below is:
12345, BA, 1, 9


Mapping Type: Separator

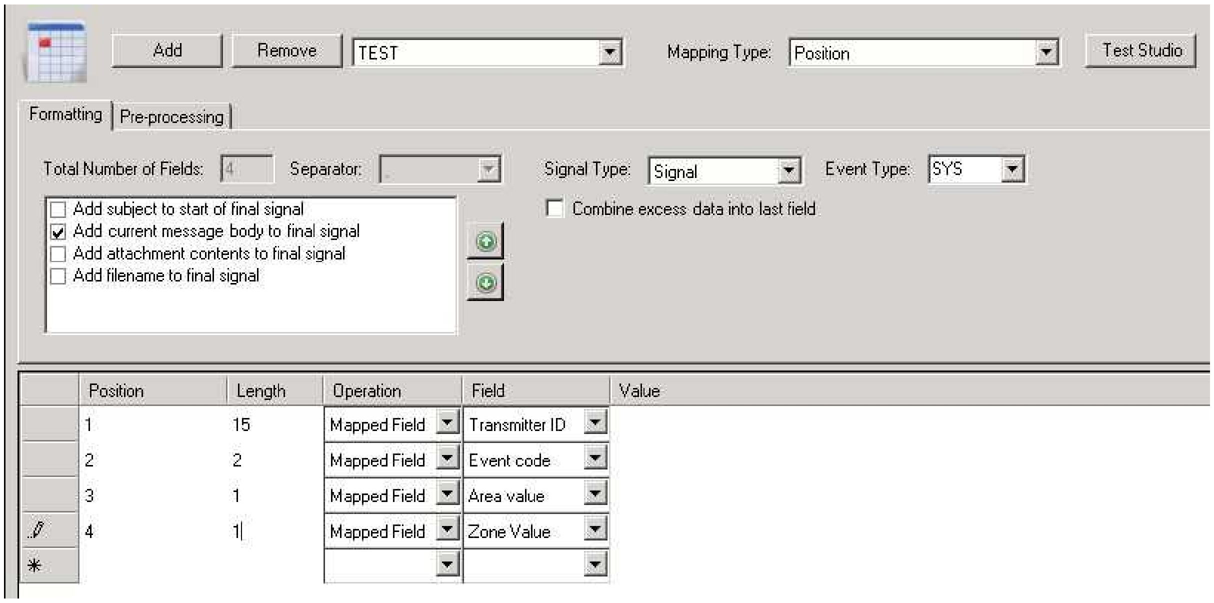
Total Number of Fields: 4 Separator: , Signal Type: Signal Event Type: SYS

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal
- [ ] Combine excess data into last field

Position	Operation	Field	Value
1	Mapped Field	Transmitter ID	
2	Mapped Field	Event code	
3	Mapped Field	Area value	
4	Mapped Field	Zone Value	

Position

The Position signal for the template below is:
12345BA19


Mapping Type: Position

Total Number of Fields: 4 Signal Type: Signal Event Type: SYS

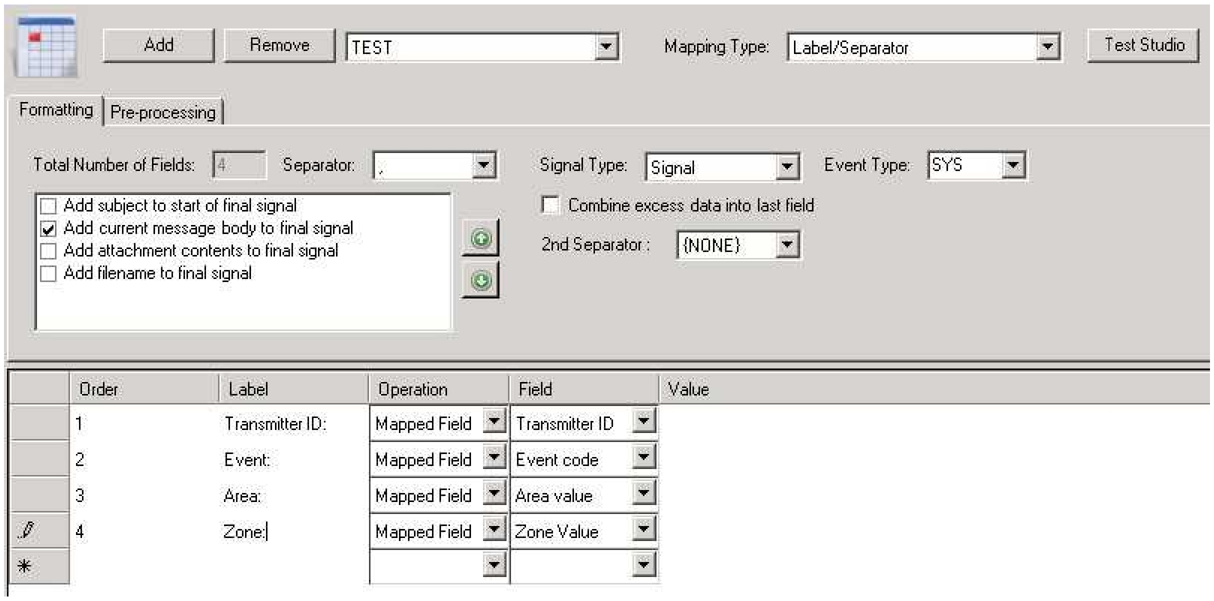
- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal
- [ ] Combine excess data into last field

Position	Length	Operation	Field	Value
1	15	Mapped Field	Transmitter ID	
2	2	Mapped Field	Event code	
3	1	Mapped Field	Area value	
4	1	Mapped Field	Zone Value	

Label/Separator

The Label/Separator signal for the template below is:
Transmitter ID:12345
Event:FA
Area:1

Zone:9



Total Number of Fields: 4 Separator: , Signal Type: Signal Event Type: SYS

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

[ ] Combine excess data into last field

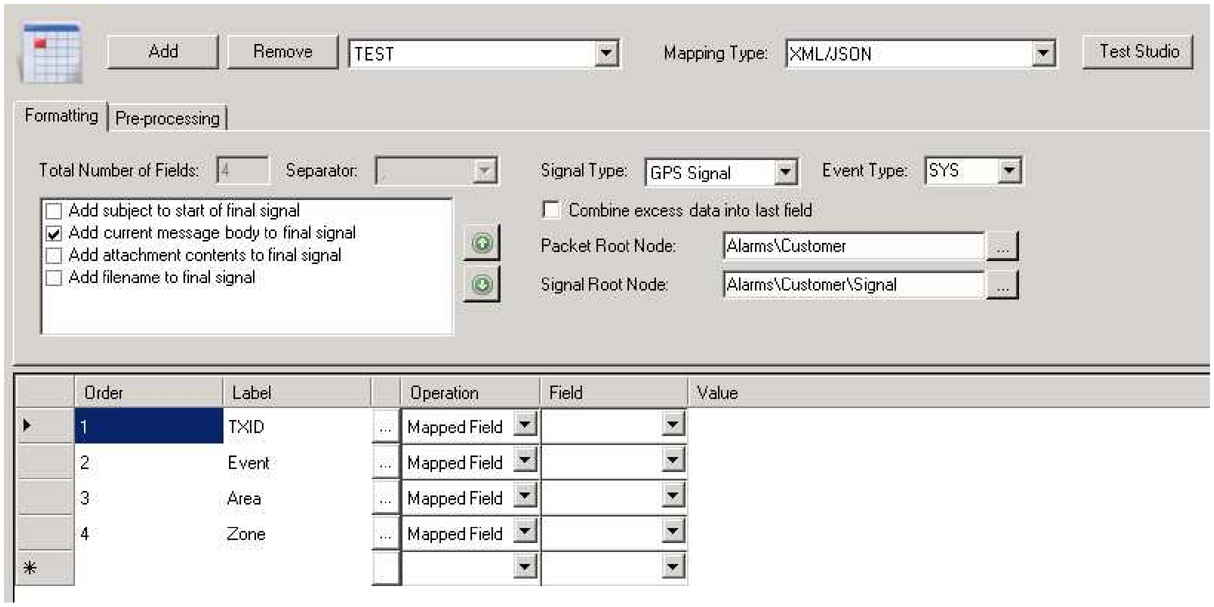
2nd Separator : {NONE}

Order	Label	Operation	Field	Value
1	Transmitter ID:	Mapped Field	Transmitter ID	
2	Event:	Mapped Field	Event code	
3	Area:	Mapped Field	Area value	
4	Zone:	Mapped Field	Zone Value	

XML

The XML signal for the template below is:

```
<?xml version="1.0"?>
<Alarms>
   <Customer TxId="12345" >
      <Signal Event="BA">
         <Zone>1</Zone>
         <Area>2</Area>
      </Signal>
   </Customer>
</Alarms>
```



TEST

Mapping Type: XML/JSON

Total Number of Fields: 4 Separator:

Signal Type: GPS Signal Event Type: SYS

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [ ] Combine excess data into last field

Packet Root Node: Alarms\Customer

Signal Root Node: Alarms\Customer\Signal

Order	Label	Operation	Field	Value
1	TXID	Mapped Field		
2	Event	Mapped Field		
3	Area	Mapped Field		
4	Zone	Mapped Field		

Pre-processing Tab

The Pre-processing tab allows you to enter a regular expression. This provides very flexible "Find and Replace" capabilities. The replacement string can serve as the separator you are using or a completely different string.

References for regular expression:

http://www.regular-expressions.info/reference.html
http://en.wikipedia.org/wiki/Regular_expression
For example, if you have a signal you want to process in a comma separated format, but each item of data is preceded by a information label, such as:

txid:12345,event:Burglary,area:2,zone:5

Using comma separation gives you four fields as follows:
txid:12345
event:Burglary
area:2
zone:5

However, the label and data are still together. To get around this, pre-process it and change the ':' to the comma separator. You then get 8 fields. In your mapping, you can choose to ignore the labels. The signal changes to the following:
Txid,12345,Event,Burglary,Area,2,Zone,5

Event Types

Users can specify the Event Type attribute of a signal tag. The following protocols are supported:

- SIA – includes any standard SIA codes, such as BA, BR, FA, FR, etc.
- CID – includes any standard contact ID codes, such as E101, R101, E103, etc.
- SYS – includes Manitou standard system codes. Current valid system codes are as follows:

Code	Description		Code	Description
*1	GSM Link Fail		*L	Low Battery
*2	GSM No Response		*LC	Late-To-Close
*3	Land Line Link Fail		*LG	Late-To-Checkin
*4	Land Line No Response		*LO	Late-To-Open
*5	GSM Resp OK		*LS	Late-To-Start
*6	Land Line Resp OK		*LT	Late-To-Test
*7	GSM Remote Link Fail		*M	Message
*8	GSM Remote Resp OK		*MD	Missing Dual Signal
*A	Activation		*N	Cancel
*A1	Unknown Card		*NB	No Battery
*A2	Unassigned Card		*O	Open
*A3	Unauthorized Access		*OE	Early Open
*AX	Unexpected Area		*OF	Off-line

*B	Bypass		*OL	Late Open
*BA	Burglary Alarm		*ON	On-line
*C	Close		*OR	Re-Open
*CA	Call Attempts		*OU	Unscheduled Open
*CE	Early Close		*OX	Unexpected Open
*CI	Caller ID		*P	Force Arm
*CL	Late Close		*PA	Panic Alarm
*CU	Unscheduled Close		*Q	Unauthorized
*CX	Unexpected Close		*R	Restore
*D	Door Access		*RB	Bell Restore
*DF	Device Test Fail		*RO	Restore Overdue
*DO	Device Test Okay		*RX	Unexpected Restore
*DT	Device Fault		*S	Supervisory
*E	Test		*SE	Service End
*EM	Equipment Message		*SS	Service Start
*ER	Error		*ST	Status
*F	AC Loss		*T	Trouble
*FA	Fire Alarm		*TA	Trap Account
*FO	Foreign Account		*TB	Bell Trouble
*FR	Fire Restore		*TP	Tamper
*FT	Fire Trouble		*TT	Two-trip Ignore

*FX	Unexpected Fire Test	*U	Unbypass
*G	Battery OK	*V	Alarm with Audio
*H	Duress	*W	Runaway Warning
*I	User Number	*X	Auxiliary
*ID	User ID	*Y	System Restore
*J	Trouble Restore	*Z	System Alarm
*K	AC Restore		

Field Descriptions

Operations

- Mapped Field - a pre-defined field with a specified position in the signal. When connected to the data source, it recognizes the field and performs the mapping.
- Value - a field with a defined value (for e.g., *A for an activated alarm or BA for a burglary alarm)
- Manipulation - a field with an entered value manipulated into a mapped field

Field

Depending on the signal type selected, the available field choices change.

For Signals:

- Unique Row ID - the unique identifier for each row
- Transmitter ID - the Transmitter ID for the device that sent the signal
- Pseudo-DNIS - allows a map to perform DNIS line setup and map setup
- Event Code - the description of codes that Manitou uses to internally display incoming signals to Operators and others who want to view or access the data

- Area Value - the numeric value assigned to the area
- Area Description - a brief description of the area
- Zone Value - the numeric value assigned to the zone
- Sensor Value - a brief description of the zone
- User Number - a numeric value assigned to the user
- User Info - brief information about the user (for e.g., the first and last name)
- Point ID - detailed information about the location, user, or account as defined in Manitou
- Minutes Ago - the number of minutes since the last signal was sent
- Date/Time (if in past) - the date and time at which the signal was originally received

For GPS:

- Unique Row ID - the number of the row of the table from which the signal was taken
- Transmitter ID - the Transmitter ID of the device that sent the signal
- Pseudo-DNIS - allows a map to DNIS line setup and map setup
- Event Code - the description of codes that Manitou uses to internally display incoming signals to Operators and others who may view or access the data
- Latitude - the latitude of the location from which the signal was sent
- Longitude - the longitude of the location from which the signal was sent
- Direction - the direction of the signal
- Degrees - degrees of the signal location
- Minutes, Seconds, Tenths - GPS indicators for plotting the position
- Speed - how fast the vehicle is traveling
- Heading - North, South, East, West, etc.
- Power - the signal strength for the Transmitter
- Comment - the signal comment

For Telemetry:

- Unique Row ID - the number of the row of the table from which the signal was taken
- Transmitter ID - the Transmitter ID from which the signal was sent
- Pseudo-DNIS - allows a map to DNIS line setup and map setup
- Event Code - the description of codes that Manitou uses to internally display incoming signals to Operators and others who want to view or access the data
- Sensor Value - a numeric value for a sensor which has been set up for the account in Manitou
- Minutes Ago - the number of minutes since the signal was sent
- Unit - components used to measure an item (for e.g., time can be measured in seconds, minutes, and hours, etc.)

- Value - the number of items reported by a measurement-based device

Connector Setup

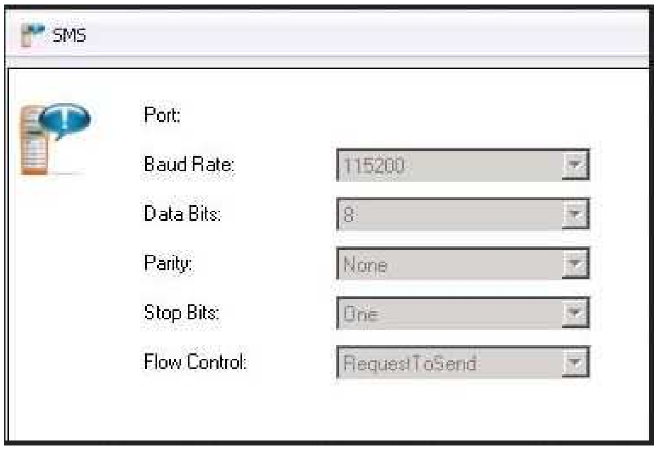
UniversalConnector currently supports the following connectors. Additional connectors are available in the MediaGateway 2. However, only those supported through the UniversalConnector are discussed in this section.

- SMS (modem)
- Email
- ODBC (Database)
- FTP
- TCP/UDP
- File
- RSS

Note: each connector must be individually licensed. Only licensed connectors are available for setup. The initial setup of any available connectors is handled by the Bold Implementation team. Connectors may be licensed at any time through the proper licensing channels.

SMS

The SMS Connector allows you to connect to an SMS modem. Both GPRS and CDMA modems are supported. When configured, the SMS connector setup displays in the c onnector section of the UniversalConnector.



Port:

Baud Rate: 115200

Data Bits: 8

Parity: None

Stop Bits: One

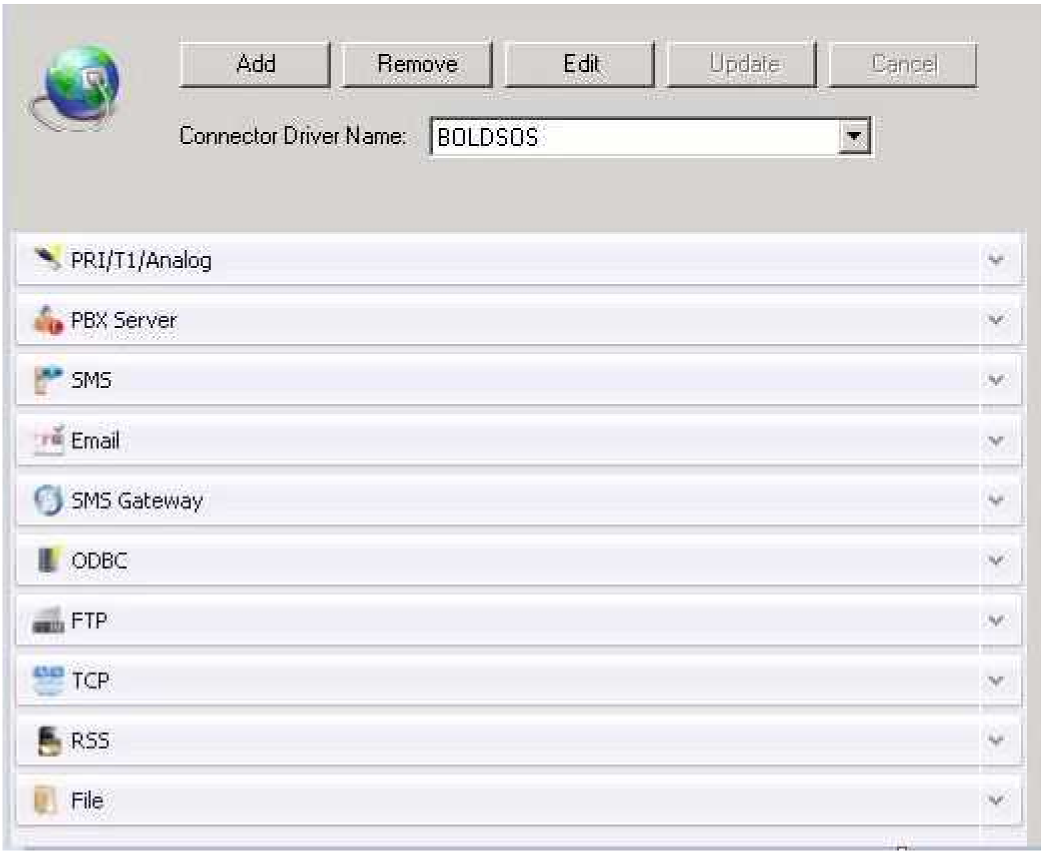
Flow Control: RequestToSend

SMS configuration

Adding an SMS Connector

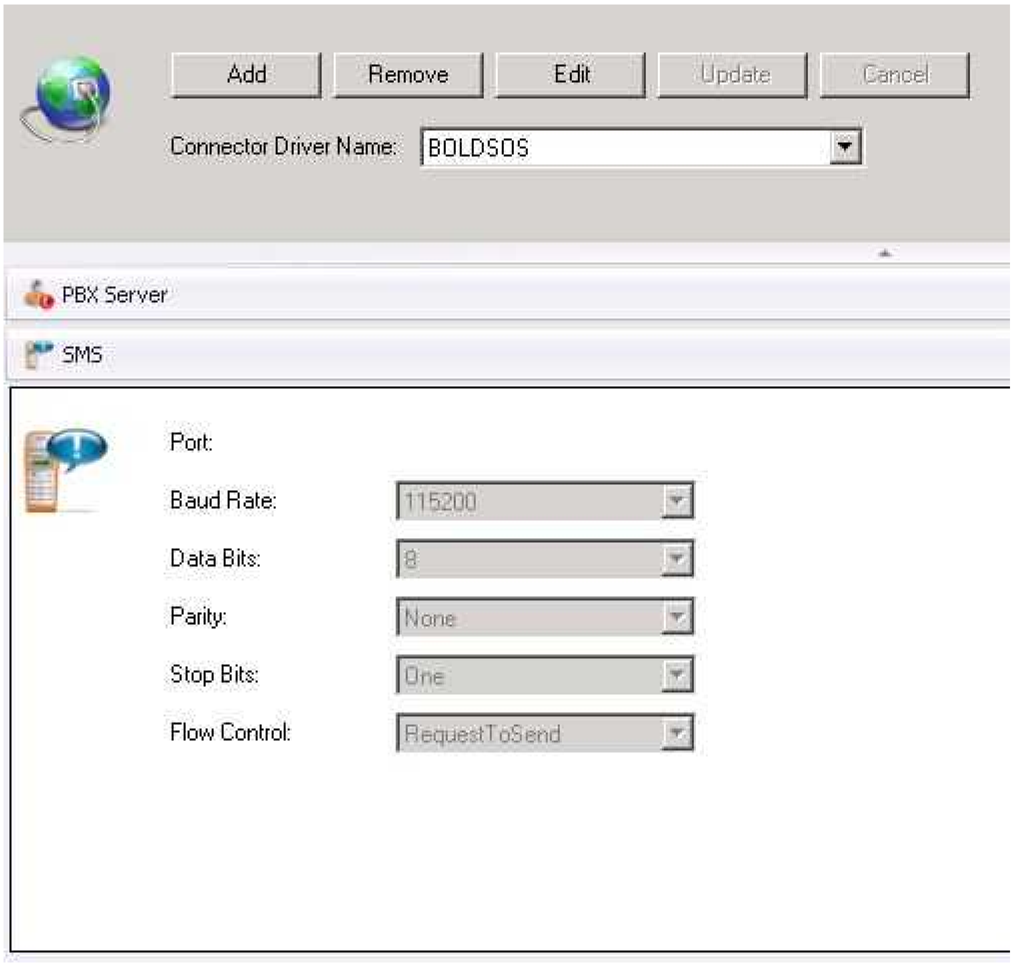
Perform the following steps to add an SMS Connector to the UniversalConnector:

1. Click "Connector" under the UniversalConnector section of the Node Tree.
Result: the following window displays:



Connector Driver Name: BOLDSOS

2. Click the down arrow to expand the SMS area of the window as shown in the following screenshot:



Connector Driver Name: BOLDSOS

Port:

Baud Rate: 115200

Data Bits: 8

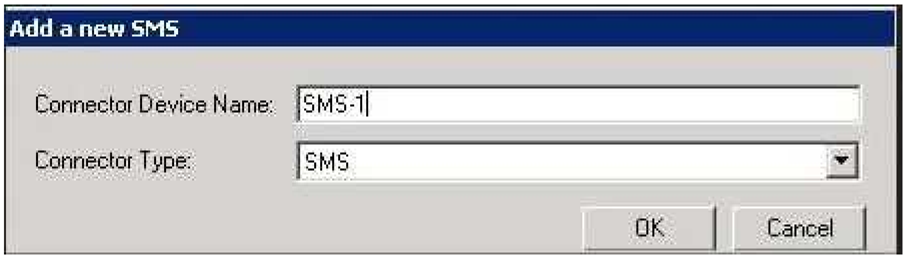
Parity: None

Stop Bits: One

Flow Control: RequestToSend

3. Click "**Add**" at the top of the window.

Result: the "Add a new SMS" dialog box displays as shown in the following screenshot:



Connector Device Name: SMS-1

Connector Type: SMS

3. Enter a device name into the "Connector Device Name:" field.
4. Confirm that the "Connector Type:" field displays as "SMS". If it does not, select "SMS" from the dropdown list.
5. Click "**OK**".

Result: the "Add a new SMS" window closes and the fields on the SMS area of the window

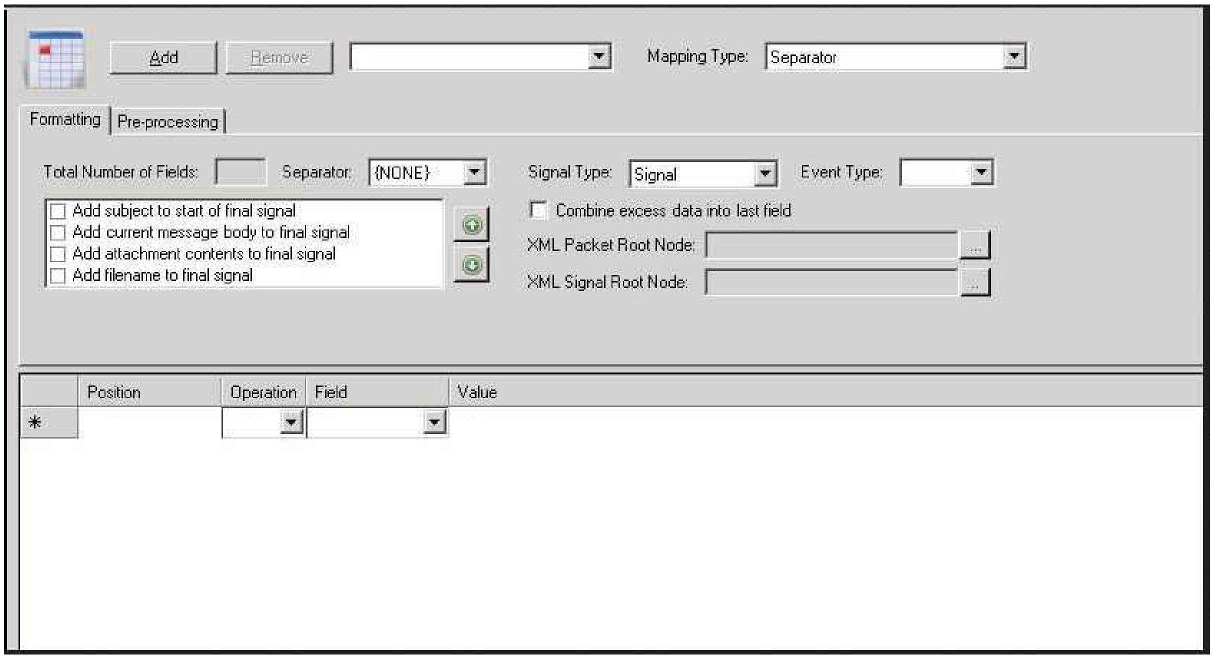
now display as populated with the default settings.
Note: you may need to modify the default settings depending on the hardware components you have installed. Use the dropdown arrows to access additional selections available for each field.

SMS Data Mapping

In order for the communication to be properly identified and submitted to Manitou, you must define the correct data mapping in the Data Mapping form.

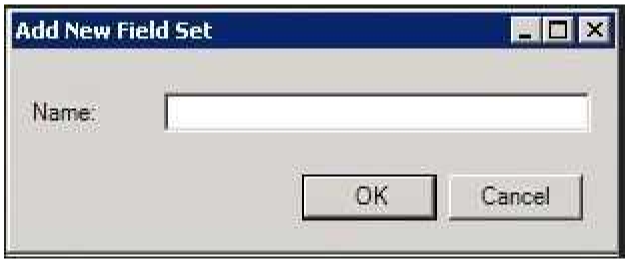
Perform the following steps to define the correct data mapping:

1. Click "Data Mapping" under UniversalConnector on the Node Tree.
Result: the Data Mapping form displays as shown in the following screenshot:

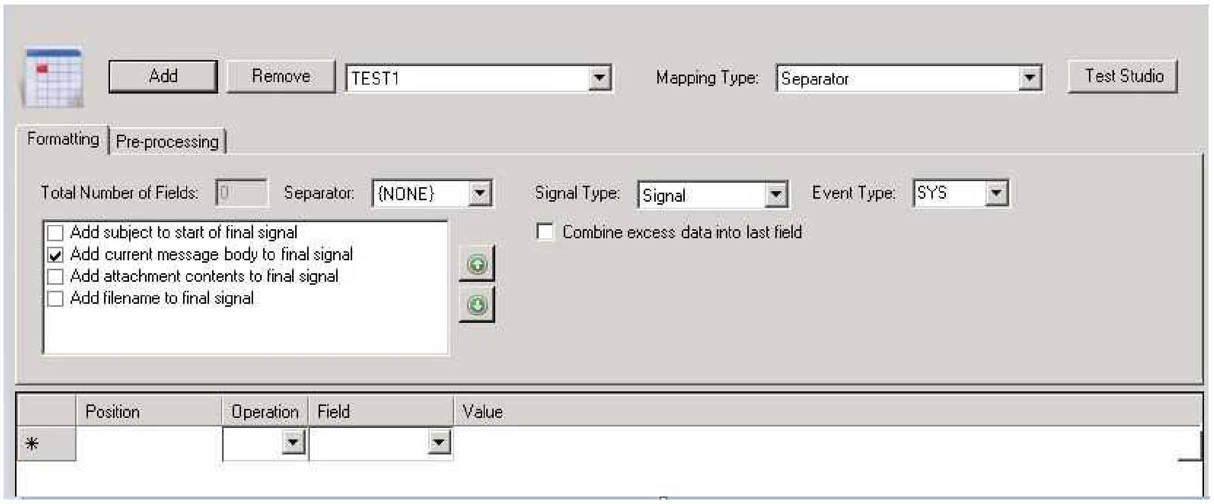


Total Number of Fields:
Separator: {NONE}
Signal Type: Signal
Event Type:
Mapping Type: Separator
Add subject to start of final signal
Add current message body to final signal
Add attachment contents to final signal
Add filename to final signal
Combine excess data into last field
XML Packet Root Node:
XML Signal Root Node:

2. Select "SMS" from the dropdown menu directly to the right of the "Remove" button.
3. Click "**Add**".
Result: the "Add New Field Set" dialog displays as shown in the following screenshot:

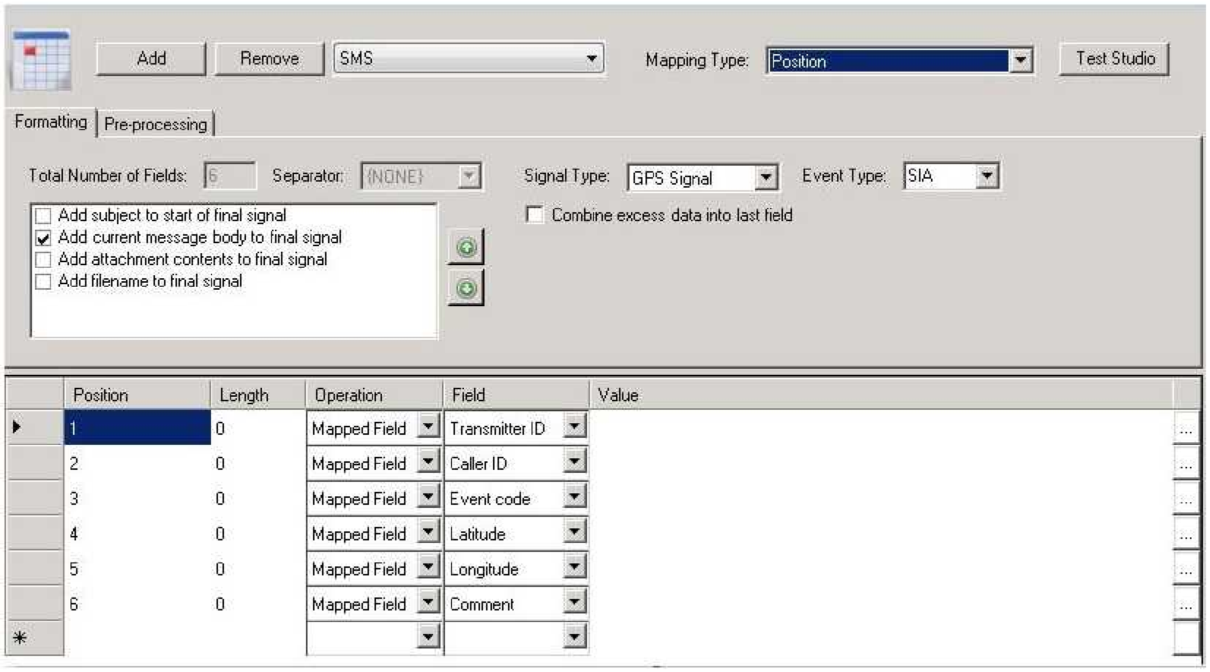


4. Enter a name into the "Name:" field, and then click "**OK**".
Result: the "Add New Field Set" window closes and the system returns you to the Data Mapping form as shown in the following screenshot:



TEST1
Mapping Type: Separator
Total Number of Fields: 0
Separator: (NONE)
Signal Type: Signal
Event Type: SYS
☐ Add subject to start of final signal
☑ Add current message body to final signal
☐ Add attachment contents to final signal
☐ Add filename to final signal
☐ Combine excess data into last field

5. Select an option from the "Mapping Type:" dropdown menu.
Note: your options are: Separator, Position, Label/Separator, and XML/JSON.
6. Select an option from the "Separator:" dropdown menu.
7. Select an option from the "Signal Type:" dropdown menu.
Note: your options are: Signal, GPS Signal, and Telemetry. For more information on these signal types, please refer to the document entitled "Field Descriptions".
8. Select an option from the "Event Type:" dropdown menu.
Note: your options are: SYS, SIA, and CID.
9. If you want to add part of the message to the final signal, select any of the four checkboxes.
10. If you want, select the "Combine excess data into the last field" checkbox.
11. In the Data Parameters section in the bottom portion of the screen, designate the order you want the information to be parsed into Manitou in the "Position" column.



SMS

Mapping Type: Position

Total Number of Fields: 6 Separator: {NONE} Signal Type: GPS Signal Event Type: SIA

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal
- [ ] Combine excess data into last field

Position	Length	Operation	Field	Value
1	0	Mapped Field	Transmitter ID	
2	0	Mapped Field	Caller ID	
3	0	Mapped Field	Event code	
4	0	Mapped Field	Latitude	
5	0	Mapped Field	Longitude	
6	0	Mapped Field	Comment	

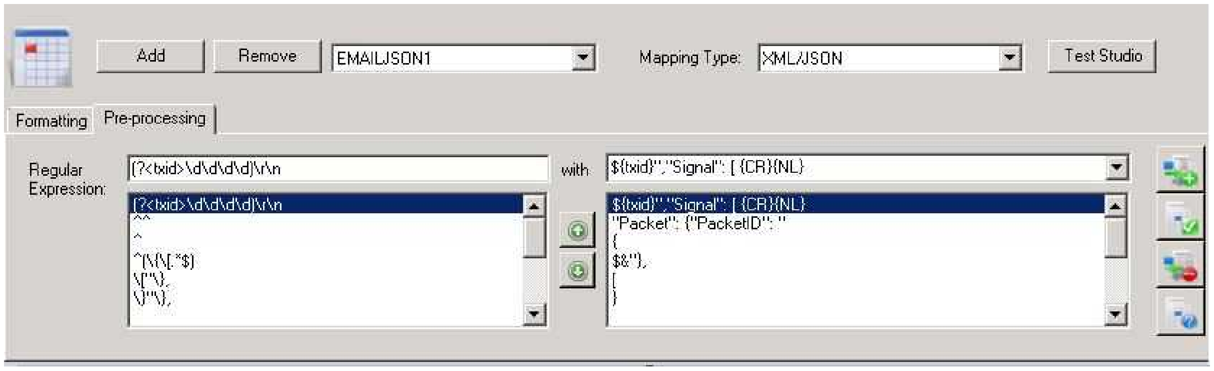
Note: the column labels in the Data Parameters area change depending on the Mapping Type you selected.

Defining Expressions in the Pre-processing Tab

The Pre-processing tab enables you to anticipate and prevent some common processing issues before they occur. For example, if a comma was designated as the separator on the Formatting tab, but there are instances where a semi-colon might come through the received signal as the separator, you can avoid this issue before it occurs.

Perform the following steps to use the Pre-processing tab to avoid common processing issues:

1. Click the Pre-processing tab on the Data Mapping form.
Result: the Pre-processing tab displays as shown in the following screenshot:



EMAILJSON1

Mapping Type: XML/JSON

Regular Expression: (?<txid>\d\d\d\d)\r\n

with ${txid}","Signal": [{CR}{NL}

```
(?<txid>\d\d\d\d)\r\n
^^
^
^(\{\[.*$)
\["\},
\}"\},
```

```
${txid}","Signal": [ {CR}{NL}
"Packet": {"PacketID": "
{
$&"},
[
}
```

2. In the "Regular Expression:" field, enter the symbol or letters used as a common expression in the signals you receive.
3. Enter the interpretation option you want for the common expression in the right-hand field.

The buttons on the right-hand side of the window perform the following functions:

Add regular expression

Update regular expression

Remove regular expression

Test Studio

Line Drivers

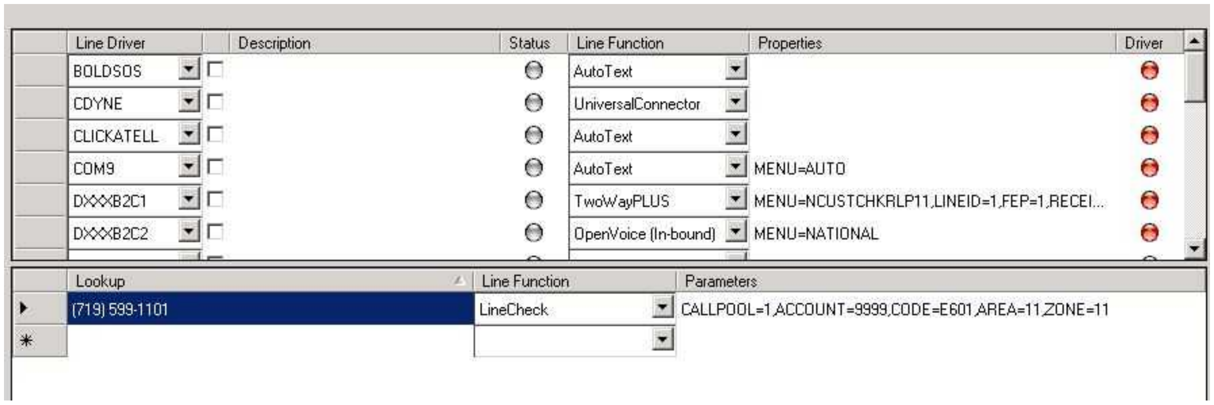
Once you have created an SMS Connector and mapped it, you must format the Line Drivers.

Note: the FEP and associated Receivers must be correctly installed and configured prior to formatting Line Drivers. To check the installation of the FEPs and Receivers, please refer to the appropriate Supervisor Workstation manual for your version of Manitou.

Perform the following steps to format the Line Drivers:

1. Click "Line Drivers" under Administration in the Node Tree.

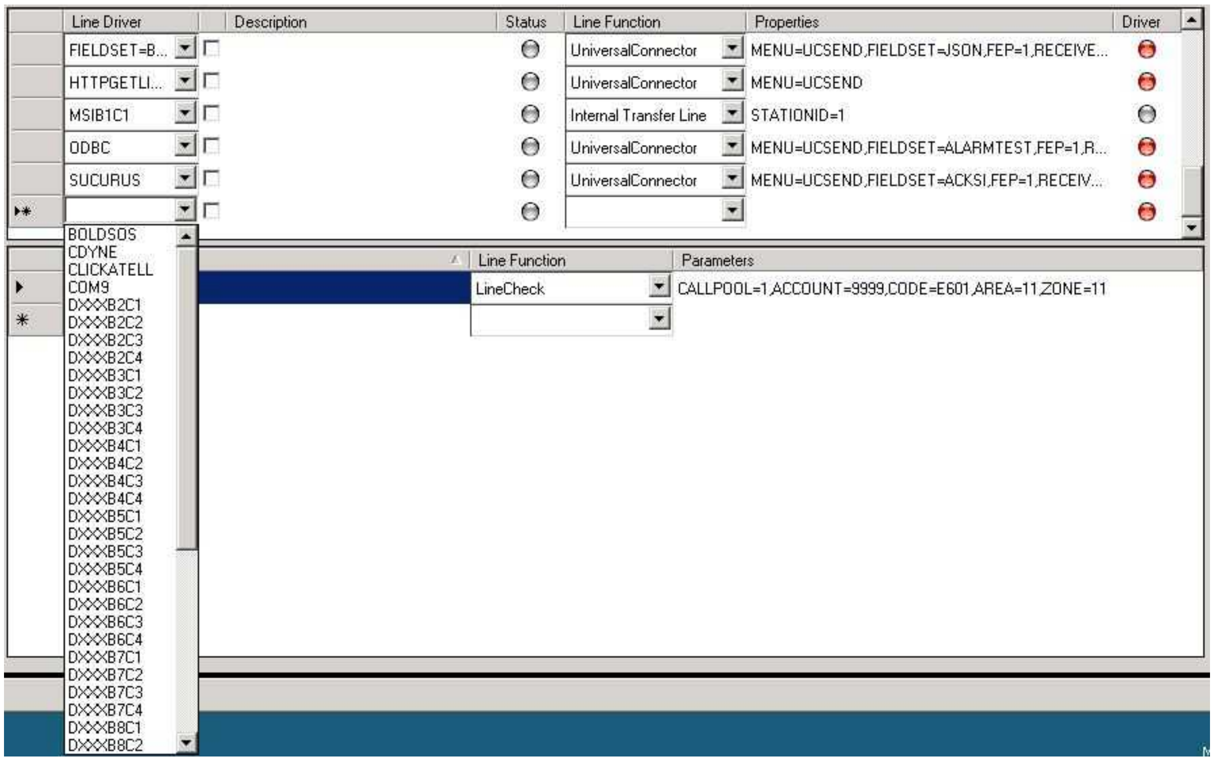
Result: the Line Drivers form displays as shown in the following screenshot:



Line Driver		Description	Status	Line Function	Properties	Driver
BOLDSOS	☐			AutoText		
CDYNE	☐			UniversalConnector		
CLICKATELL	☐			AutoText		
COM9	☐			AutoText	MENU=AUTO	
DXXXB2C1	☐			TwoWayPLUS	MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	
DXXXB2C2	☐			OpenVoice (In-bound)	MENU=NATIONAL	

	Lookup	Line Function	Parameters
▸	(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*			

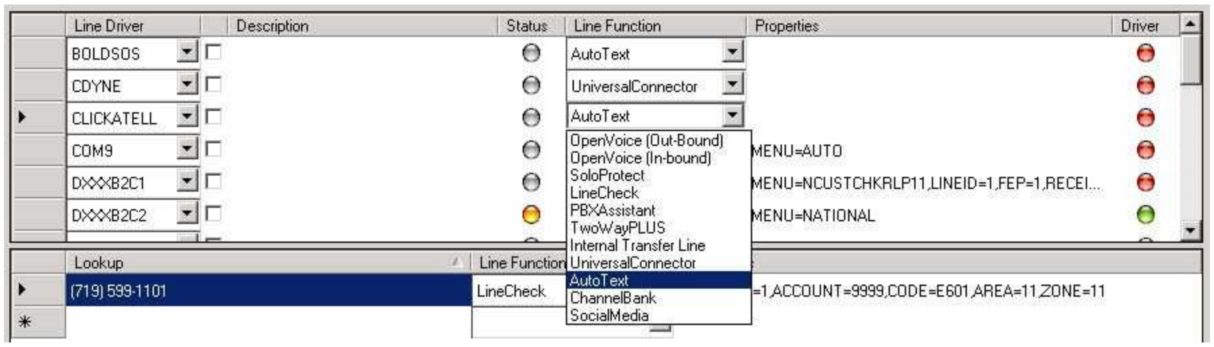
2. In the "Line Driver" column, select the Line Driver from the pulldown list in the next open row (indicated with an asterisk) as shown in the following screenshot:



	Line Driver		Description	Status	Line Function	Properties	Driver
	FIELDSET=B...	☐			UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	
	HTTPGETLI...	☐			UniversalConnector	MENU=UCSEND	
	MSIB1C1	☐			Internal Transfer Line	STATIONID=1	
	ODBC	☐			UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	
	SUCURUS	☐			UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	
▸*		☐					

	Line Function	Parameters
▸	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*		

3. If you want the Line Driver to be designated as active, select the checkbox to the right of it. If you do not want it to be active, leave the checkbox deselected.
4. Enter a description into the "Description" field.
5. Select the type of "Line Function" you want from the pulldown menu as shown in the following screenshot:

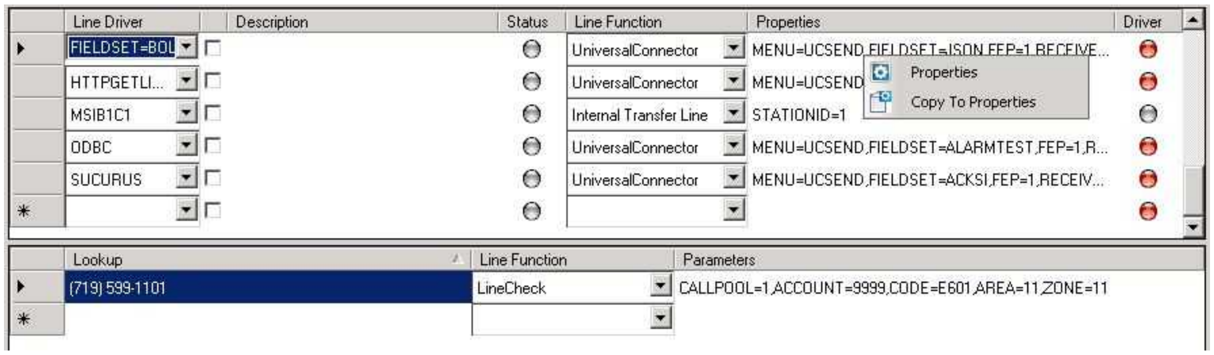


Line Driver	Description	Status	Line Function	Properties	Driver
BOLDSOS	☐	●	AutoText		●
CDYNE	☐	●	UniversalConnector		●
CLICKATELL	☐	●	AutoText		●
COM9	☐	●		MENU=AUTO	●
DXXXB2C1	☐	●		MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	●
DXXXB2C2	☐	●		MENU=NATIONAL	●

Lookup	Line Function	Parameters
(719) 599-1101	LineCheck	=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*		

Establish Properties

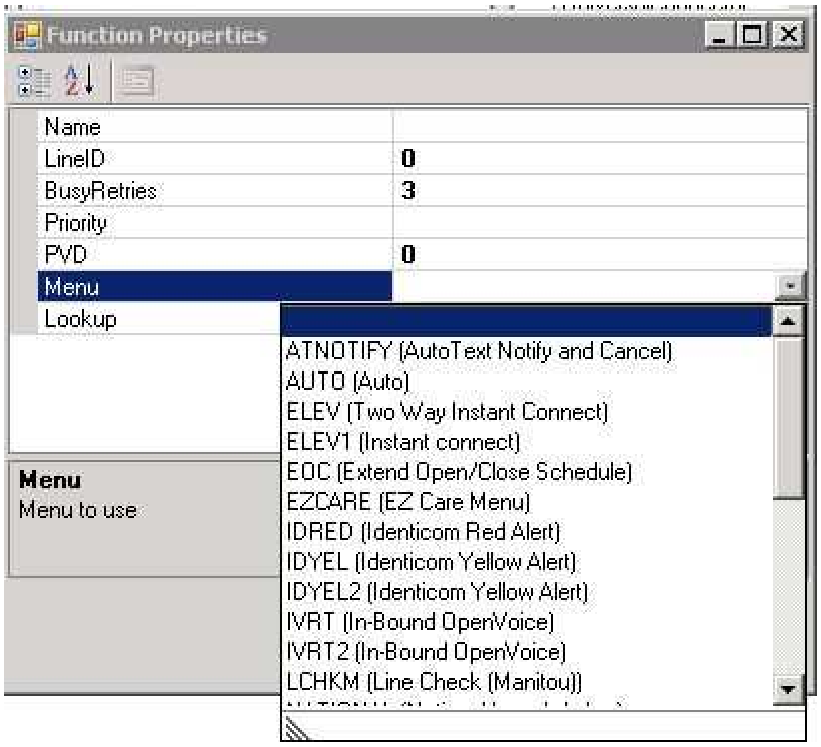
1. To enter property information for a Line Driver, select the Driver you want. Then, right-click in the "Properties" column and select Properties from the menu that displays. You also have the option of copying properties from an already existing Line Driver by selecting the "Copy to Properties" option as shown in the following screenshot:



Line Driver	Description	Status	Line Function	Properties	Driver
FIELDSET=BOL	☐	●	UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	●
HTTPGETLI...	☐	●	UniversalConnector	MENU=UCSEND	●
MSIB1C1	☐	●	Internal Transfer Line	STATIONID=1	●
ODBC	☐	●	UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	●
SUCURUS	☐	●	UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	●
*	☐	●			●

Lookup	Line Function	Parameters
(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*		

Result: the Function Properties window displays as shown in the following screenshot:



Name	
LineID	0
BusyRetries	3
Priority	
PVD	0
Menu	
Lookup	

Menu
Menu to use

ATNOTIFY (AutoText Notify and Cancel)
AUTO (Auto)
ELEV (Two Way Instant Connect)
ELEV1 (Instant connect)
EOC (Extend Open/Close Schedule)
EZCARE (EZ Care Menu)
IDRED (Identicom Red Alert)
IDYEL (Identicom Yellow Alert)
IDYEL2 (Identicom Yellow Alert)
IVRT (In-Bound OpenVoice)
IVRT2 (In-Bound OpenVoice)
LCHKM (Line Check (Manitou))

Note: which function list items display on the Function Properties menu depend on the Line Driver you selected.

2. Enter the values you want in the right column to correspond to the function list on the left. Some functions have dropdown menus (as shown in the screenshot above). To access a dropdown menu, click at the edge of the right column until a down arrow displays. Click the arrow, and scroll through the list to select the item you want.
3. Click "**OK**" when finished, and the click "**Save**".

SMS Example

SMS Signal

The SMS Connector uses caller ID as the Transmitter ID. Then, it identifies the Event, Point ID, and zone based on definitions established in the Data Mapping section of the MediaGateway 2. The following is an example of an SMS signal as it is interpreted through the UniversalConnector.

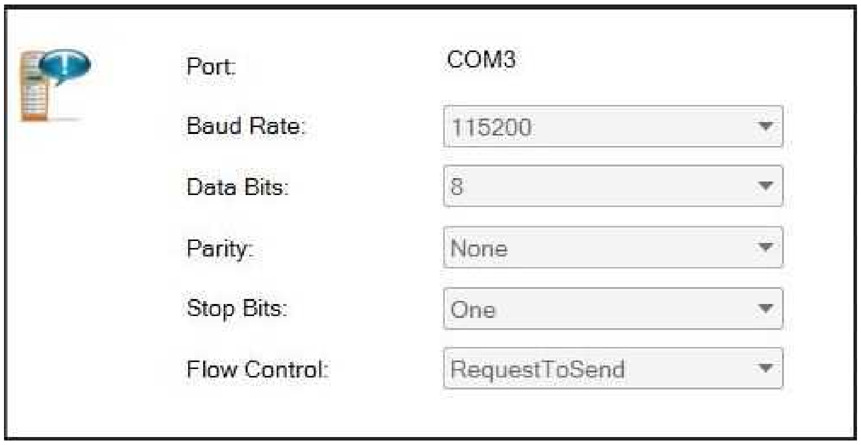
Signal example: Text message - S1, BA, Kitchen, 1

In the above example, the UniversalConnector is capable of correctly identifying the following signal pieces:

- S1 - Fieldset Identifier
- BA - Event (Burglary Alarm)
- Kitchen - Point ID
- 1 - Zone

Connector

In the above example, the SMS Connector configuration should match the following screenshot:

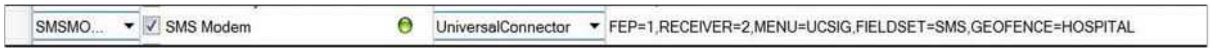


Port:	COM3
Baud Rate:	115200
Data Bits:	8
Parity:	None
Stop Bits:	One
Flow Control:	RequestToSend

SMS Connector Setup example

Line Maps

The screenshot below shows the line mapping for the SMS modem. Notice that the Fieldset has been identified.

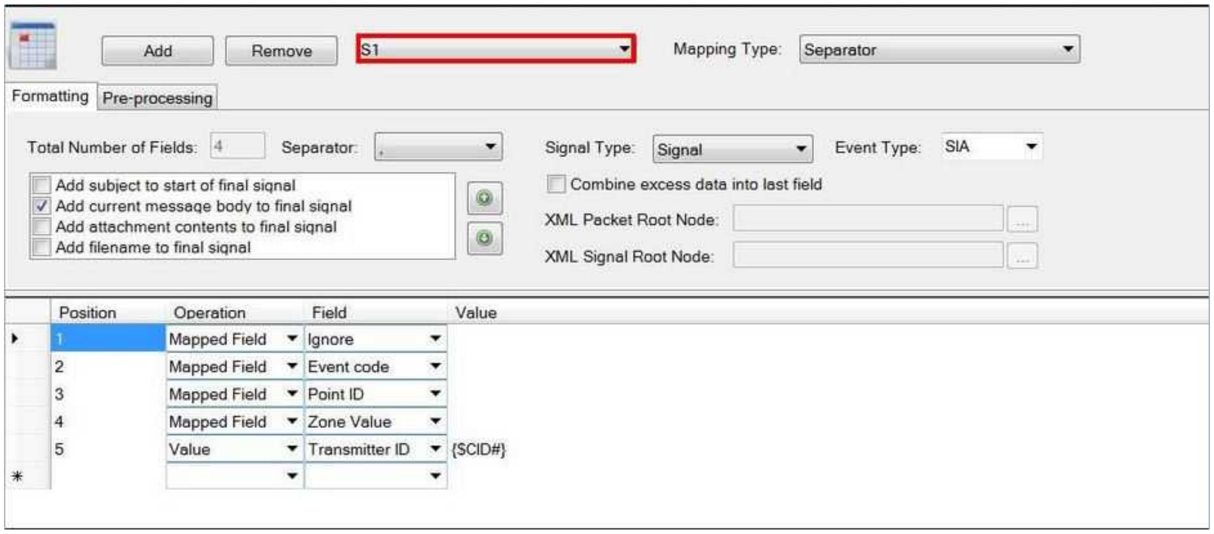


SMSMO... ☑ SMS Modem ● UniversalConnector FEP=1,RECEIVER=2,MENU=UCSIG,FIELDSET=SMS,GEOFENCE=HOSPITAL

SMS Line Maps example

Data Mapping

In this example, the line map first points to the SMS Fieldset:



S1

Mapping Type: Separator

Total Number of Fields: 4
Separator: ,
Signal Type: Signal
Event Type: SIA

- [ ] Add subject to start of final signal
- [✓] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

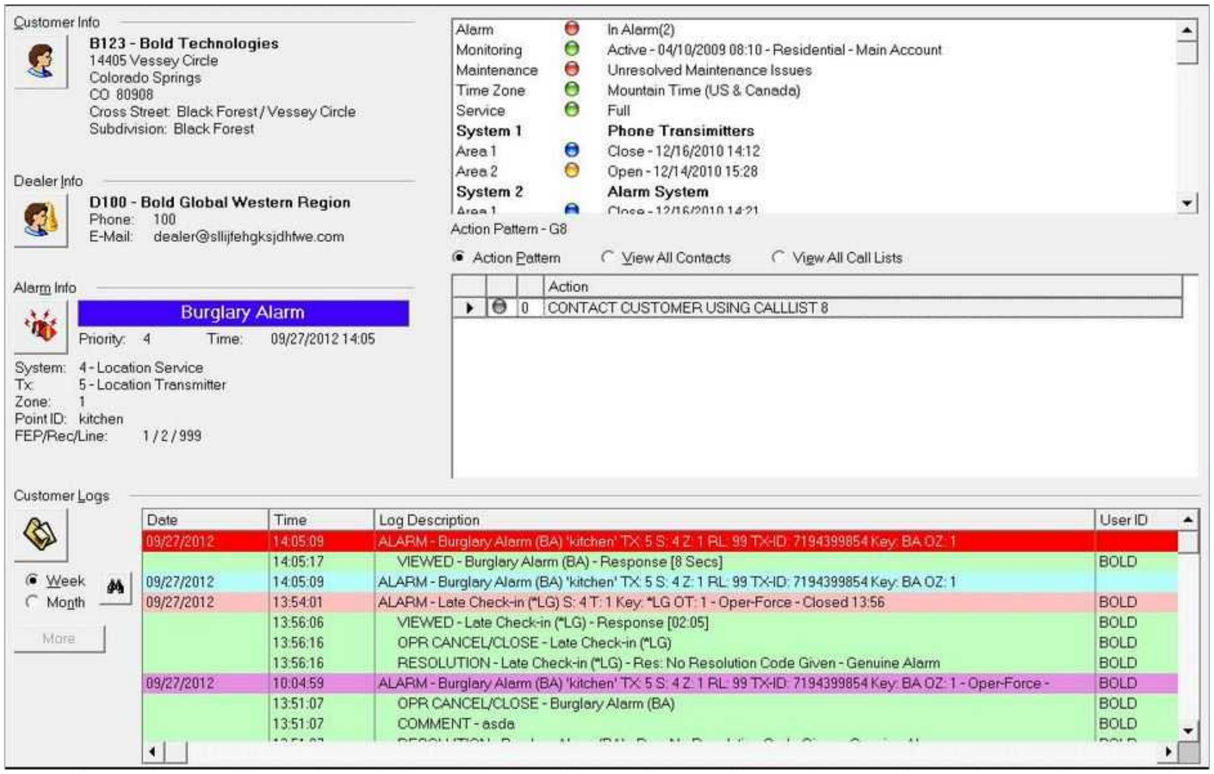
- [ ] Combine excess data into last field

XML Packet Root Node:
XML Signal Root Node:

Position	Operation	Field	Value
1	Mapped Field	Ignore	
2	Mapped Field	Event code	
3	Mapped Field	Point ID	
4	Mapped Field	Zone Value	
5	Value	Transmitter ID	{$CID#}

Manitou Alarm

When the signal is sent into Manitou, the following alarm is created:


Customer Info

B123 - Bold Technologies
14405 Vessey Circle
Colorado Springs
CO 80908
Cross Street: Black Forest / Vessey Circle
Subdivision: Black Forest

Dealer Info

D100 - Bold Global Western Region
Phone: 100
E-Mail: dealer@sllijfehgksjdhfwe.com

Alarm Info

Burglary Alarm
Priority: 4 Time: 09/27/2012 14:05
System: 4 - Location Service
Tx: 5 - Location Transmitter
Zone: 1
Point ID: kitchen
FEP/Rec/Line: 1 / 2 / 999

Alarm	🔴	In Alarm(2)
Monitoring	🟢	Active - 04/10/2009 08:10 - Residential - Main Account
Maintenance	🔴	Unresolved Maintenance Issues
Time Zone	🟢	Mountain Time (US & Canada)
Service	🟢	Full
System 1		**Phone Transmitters**
Area 1	🔵	Close - 12/16/2010 14:12
Area 2	🟡	Open - 12/14/2010 15:28
System 2		**Alarm System**
Area 1	🔵	Close - 12/16/2010 14:21

Action Pattern - G8

(● Action Pattern) ○ View All Contacts ○ View All Call Lists

			Action
▶		0	CONTACT CUSTOMER USING CALLLIST 8

Customer Logs

(● Week) ○ Month

Date	Time	Log Description	User ID
09/27/2012	14:05:09	ALARM - Burglary Alarm (BA) 'kitchen' TX: 5 S: 4 Z: 1 RL: 99 TX-ID: 7194399854 Key: BA OZ: 1	
	14:05:17	VIEWED - Burglary Alarm (BA) - Response [8 Secs]	BOLD
09/27/2012	14:05:09	ALARM - Burglary Alarm (BA) 'kitchen' TX: 5 S: 4 Z: 1 RL: 99 TX-ID: 7194399854 Key: BA OZ: 1	
09/27/2012	13:54:01	ALARM - Late Check-in (*LG) S: 4 T: 1 Key: *LG OT: 1 - Oper-Force - Closed 13:56	BOLD
	13:56:06	VIEWED - Late Check-in (*LG) - Response [02:05]	BOLD
	13:56:16	OPR CANCEL/CLOSE - Late Check-in (*LG)	BOLD
	13:56:16	RESOLUTION - Late Check-in (*LG) - Res: No Resolution Code Given - Genuine Alarm	BOLD
09/27/2012	10:04:59	ALARM - Burglary Alarm (BA) 'kitchen' TX: 5 S: 4 Z: 1 RL: 99 TX-ID: 7194399854 Key: BA OZ: 1 - Oper-Force -	BOLD
	13:51:07	OPR CANCEL/CLOSE - Burglary Alarm (BA)	BOLD
	13:51:07	COMMENT - asda	BOLD

SMS translated to Alarm in Manitou

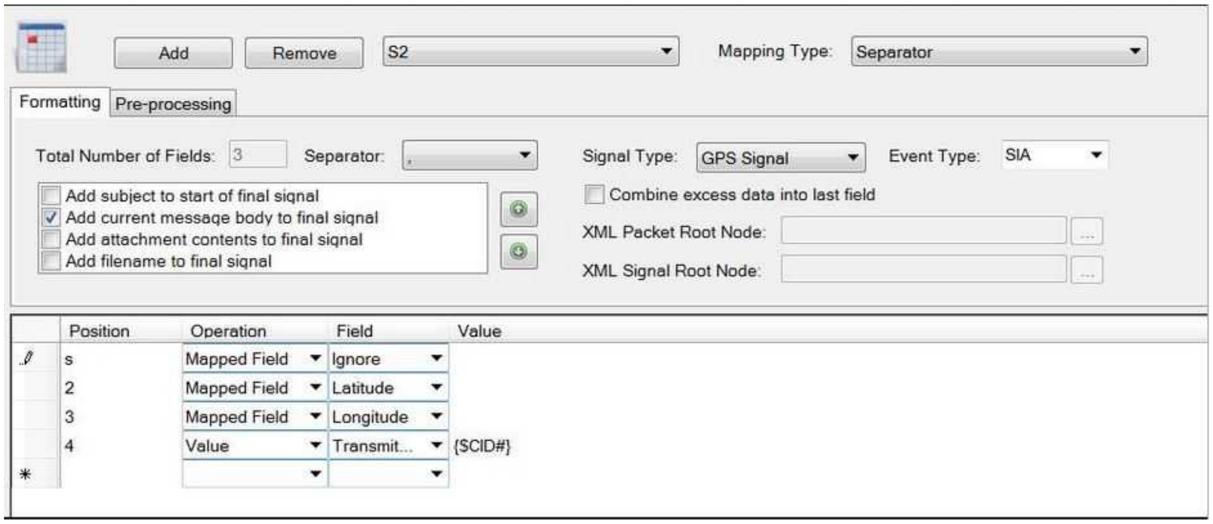
GPS Alarm

The GPS alarm is essentially the same as an SMS alarm.

GPS example:

PA alarm with GPS information, using the caller ID as a Transmitter ID and a Geo-Fence:

- S2, 38.938566,-104.717154 - Inside
- S2, 38.941286,-104.719182 – Outside



S2

Mapping Type: Separator

Total Number of Fields: 3 Separator: ,

Signal Type: GPS Signal Event Type: SIA

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [ ] Combine excess data into last field

XML Packet Root Node:

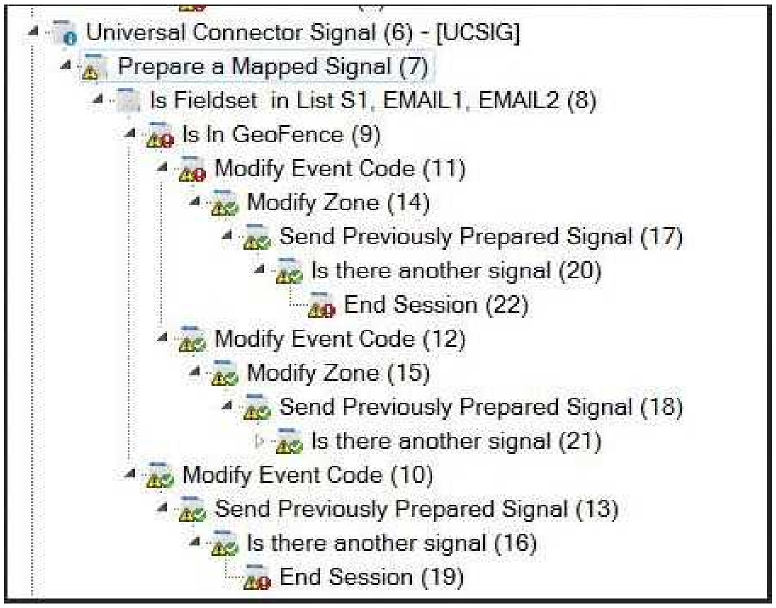
XML Signal Root Node:

Position	Operation	Field	Value
s	Mapped Field	Ignore	
2	Mapped Field	Latitude	
3	Mapped Field	Longitude	
4	Value	Transmit...	{SCID#}

Data Mapping example

Line Driver, Menu Item

The menu item assigned in the Line Driver makes all this work. The screenshot below shows the item "Prepare a Mapped Signal". The menu takes in the GPS signal, determines whether it is inside or outside the Geo-Fence, and then modifies the Event Code accordingly.

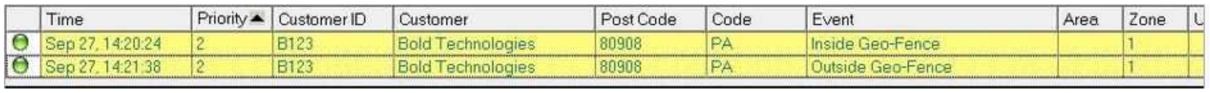


```
Universal Connector Signal (6) - [UCSIG]
  Prepare a Mapped Signal (7)
    Is Fieldset  in List S1, EMAIL1, EMAIL2 (8)
      Is In GeoFence (9)
        Modify Event Code (11)
          Modify Zone (14)
            Send Previously Prepared Signal (17)
              Is there another signal (20)
                End Session (22)
        Modify Event Code (12)
          Modify Zone (15)
            Send Previously Prepared Signal (18)
              Is there another signal (21)
      Modify Event Code (10)
        Send Previously Prepared Signal (13)
          Is there another signal (16)
            End Session (19)
```

GPS Menu Item example

Manitou Alarm

When both signals above are sent (inside and outside coordinates), two different alarms are received into Manitou. This is because one signal is coming from inside the Geo-Fence and the other is coming from outside, as seen below:

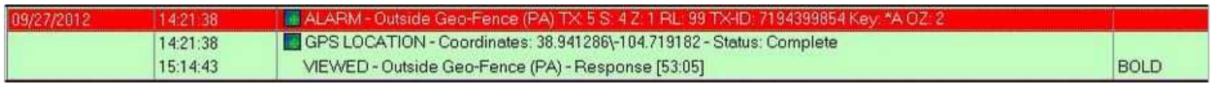


	Time	Priority	Customer ID	Customer	Post Code	Code	Event	Area	Zone	U
●	Sep 27, 14:20:24	2	B123	Bold Technologies	80908	PA	Inside Geo-Fence		1	
●	Sep 27, 14:21:38	2	B123	Bold Technologies	80908	PA	Outside Geo-Fence		1	

Alarm examples

Looking at the above outside Geo-Fence alarm, the following data is also presented:

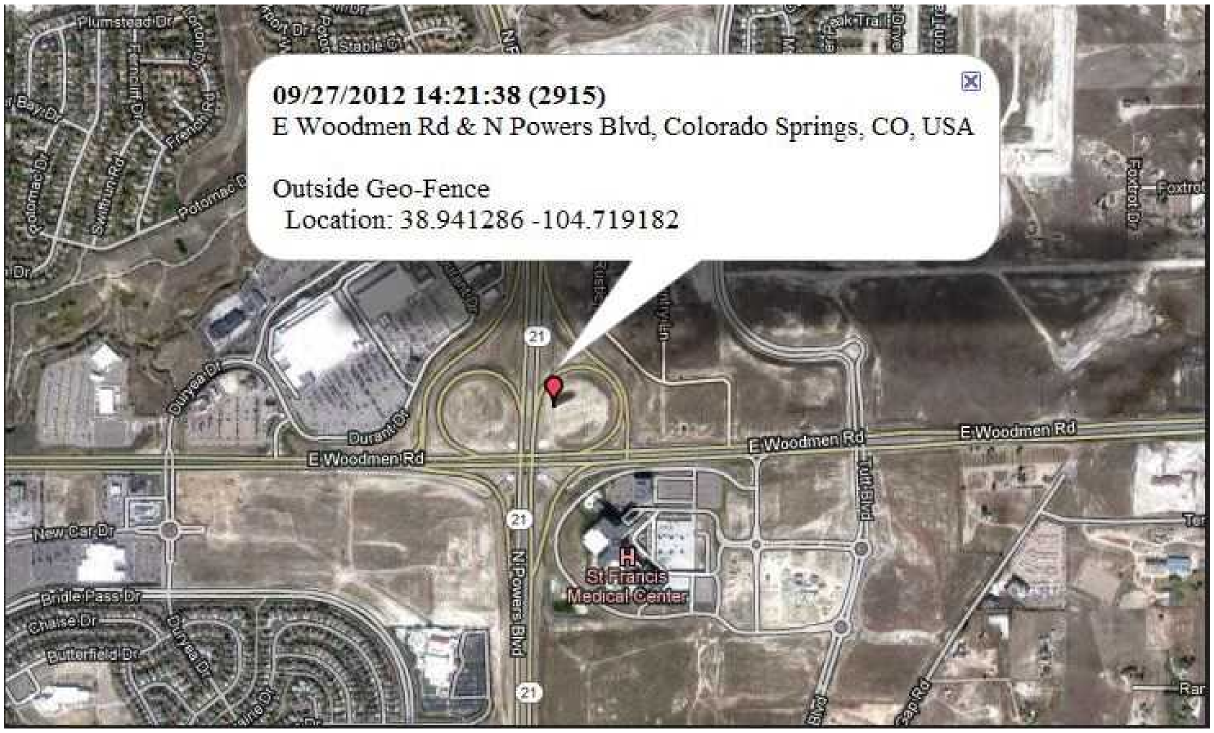
- The alarm comes into the Log as a GPS alarm.



```
09/27/2012   14:21:38   ALARM - Outside Geo-Fence (PA) TX: 5 S: 4 Z: 1 RL: 99 TX-ID: 7194399854 Key: *A OZ: 2
             14:21:38   GPS LOCATION - Coordinates: 38.941286\-104.719182 - Status: Complete
             15:14:43   VIEWED - Outside Geo-Fence (PA) - Response [53:05]                              BOLD
```

GPS Alarm example

- It also displays the map and location coordinates.



09/27/2012 14:21:38 (2915)
E Woodmen Rd & N Powers Blvd, Colorado Springs, CO, USA

Outside Geo-Fence
Location: 38.941286 -104.719182

GPS Alarm Map example

Email

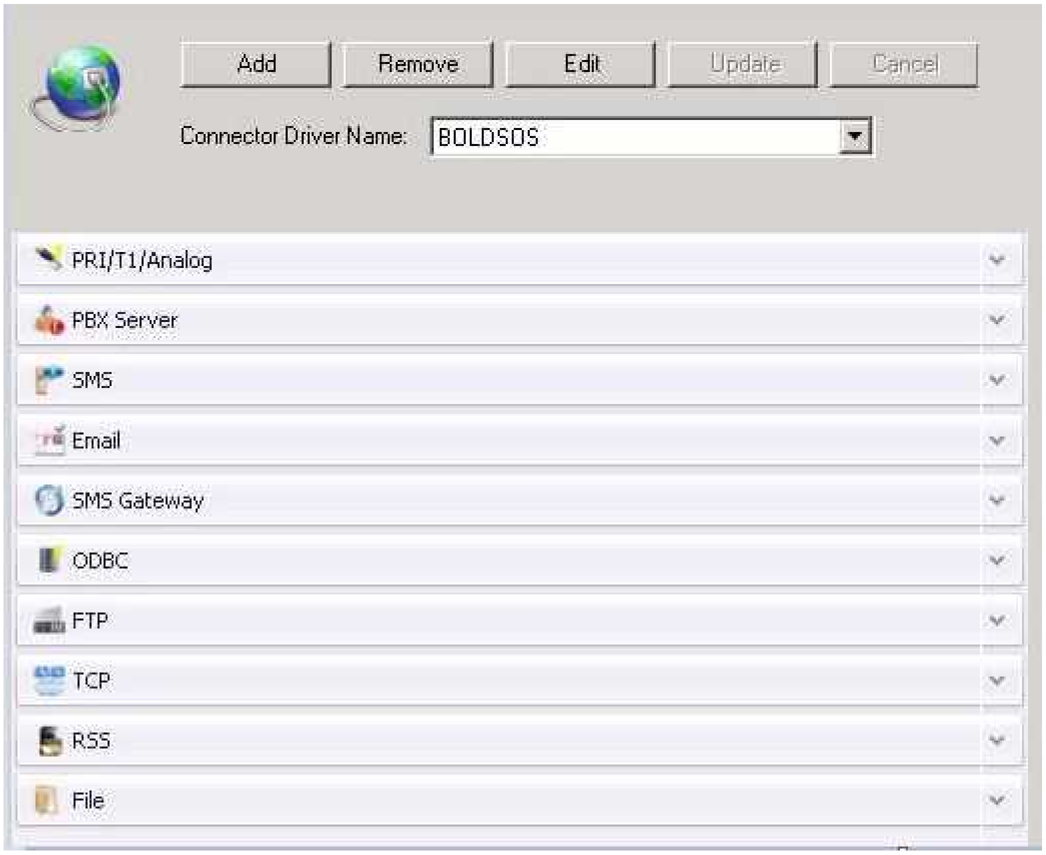
When the Email Connector is correctly configured, the MediaGateway 2 logs in and checks for a particular type of message at specific intervals. It then parses the signal and sends it into Manitou as an alarm. For example, you can configure the UniversalConnector to look in the C:/Temp file for any XML file extensions. You can also configure the Email Connector to include the subject, body, and attachments of an email with the information sent to Manitou.

When configured, the Email Connector configuration displays in the connector section under the UniversalConnector.

Adding an Email Connector

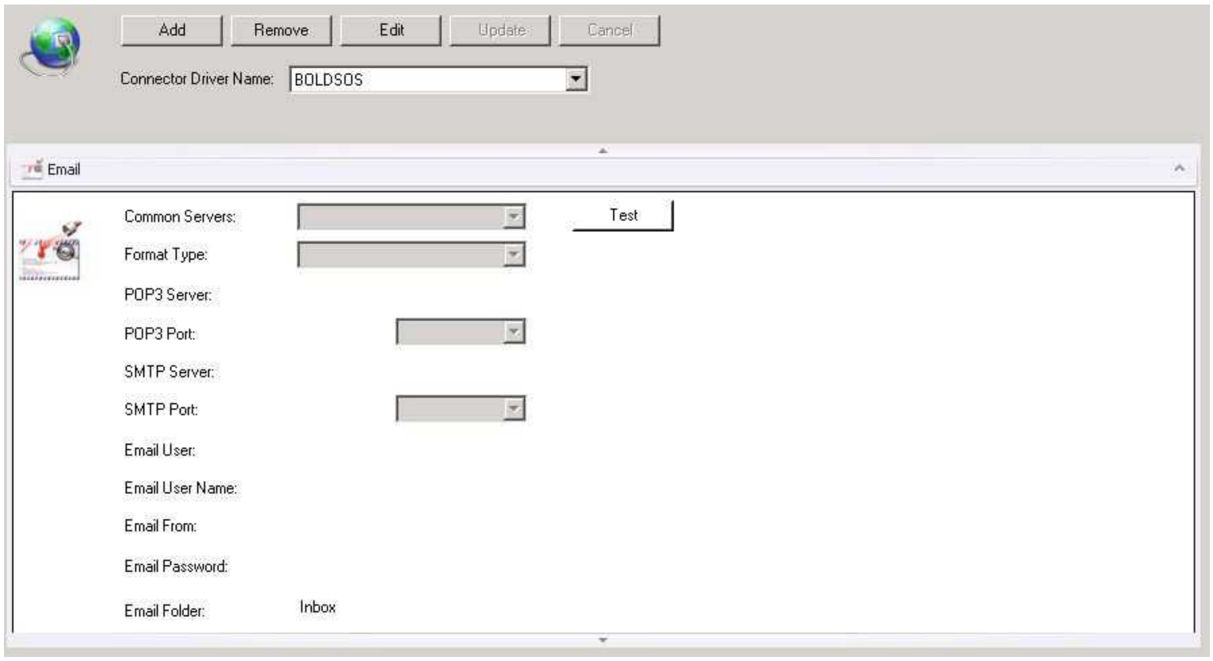
Perform the following steps to add an Email Connector:

1. Click "Connector" under the UniversalConnector section of the Node Tree.
Result: the following window displays:



Connector Driver Name: BOLDSOS

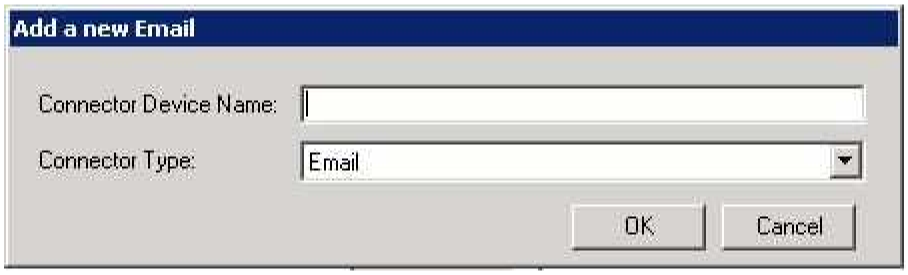
2. Click to the down arrow to expand the Email area of the window as shown in the following screenshot:



Connector Driver Name: BOLDSOS

Common Servers:
Format Type:
POP3 Server:
POP3 Port:
SMTP Server:
SMTP Port:
Email User:
Email User Name:
Email From:
Email Password:
Email Folder: Inbox

3. Click "**Add**".
Result: the "Add a new Email" dialog displays as shown in the following screenshot:



4. Enter a device name into the "Connector Device Name:" field.
5. Confirm that the "Connector Type:" field displays as "Email". If it does not, select "Email" from the dropdown list.
6. Click "**OK**".

Result: the "Add a new Email" window closes and the fields on the Email area of the window now display as populated with the default settings.
Note: you may need to modify the default settings depending on the hardware components you have installed. Use the dropdown arrows to access additional selections available for each field.

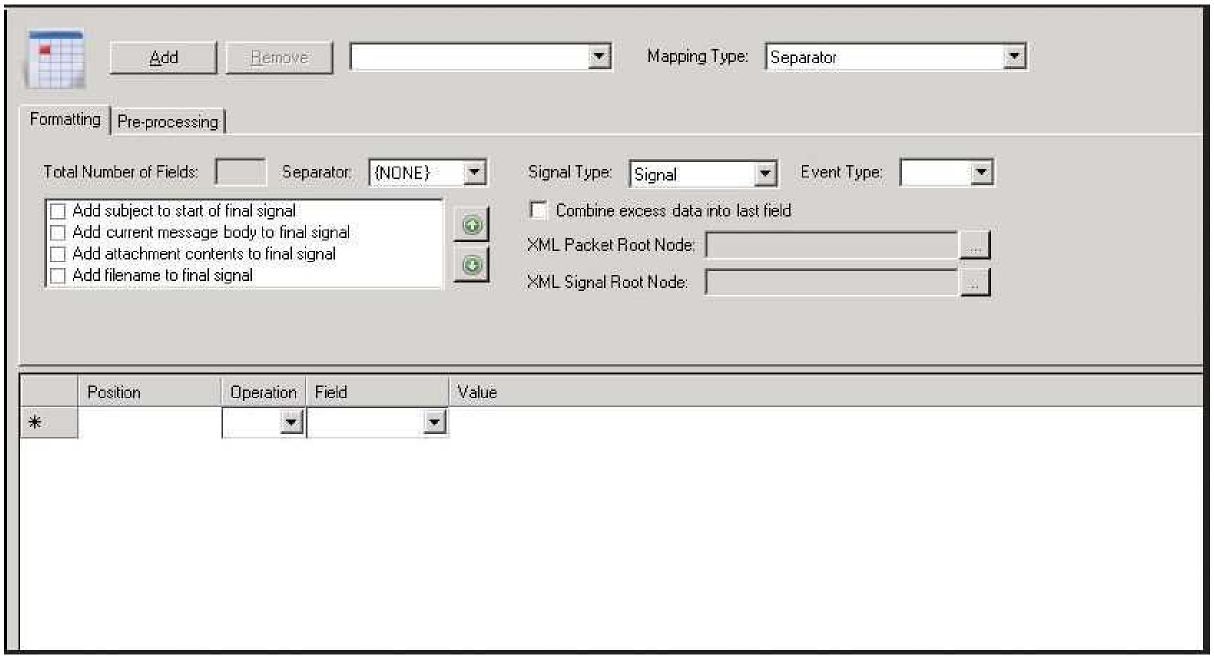
Data Mapping

In order for the communication to be properly identified and submitted to Manitou, you must define the correct data mapping in the Data Mapping form.

Perform the following steps to define the correct data mapping:

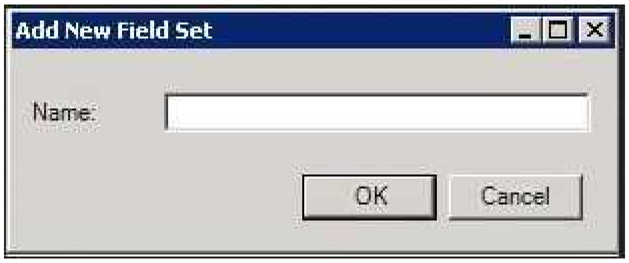
1. Click "Data Mapping" under UniversalConnector on the Node Tree.

Result: the Data Mapping form displays as shown in the following screenshot:



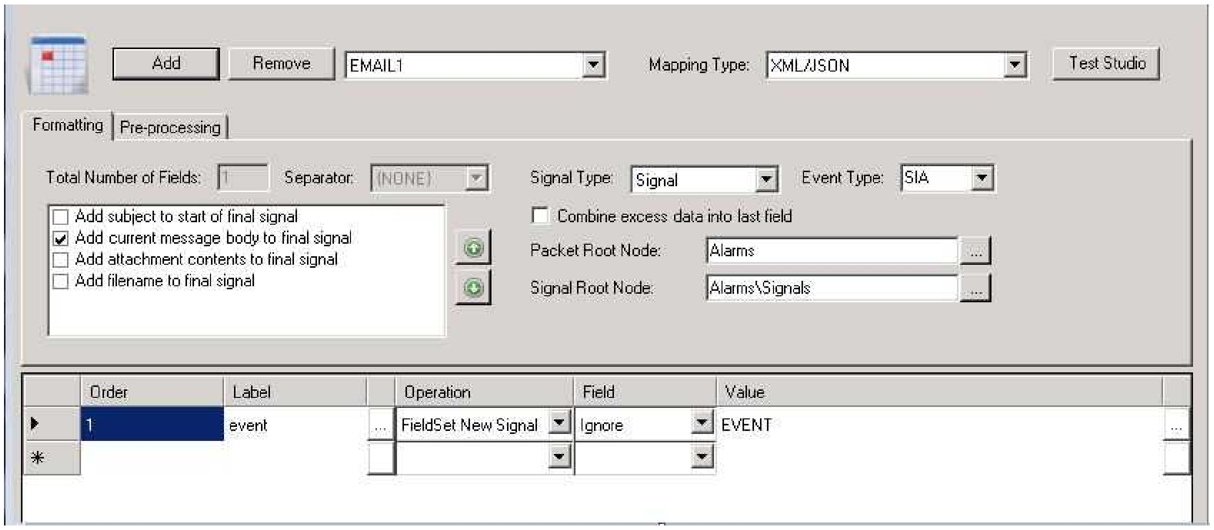
2. Select "Email" from the dropdown menu directly to the right of the "Remove" button.
3. Click "**Add**".

Result: the "Add New Field Set" dialog displays as shown in the following screenshot:



4. Enter a name into the "Name:" field, and then click "**OK**".

Result: the "Add New Field Set" window closes and the system returns you to the Data Mapping form as shown in the following screenshot:



Mapping Type: XML/JSON

Total Number of Fields: 1

Separator: (NONE)

Signal Type: Signal

Event Type: SIA

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [ ] Combine excess data into last field

Packet Root Node: Alarms

Signal Root Node: Alarms\Signals

Order	Label	Operation	Field	Value
1	event	FieldSet New Signal	Ignore	EVENT

5. Select an option from the "Mapping Type:" dropdown menu.

Note: your options are: Separator, Position, Label/Separator, and XML/JSON.

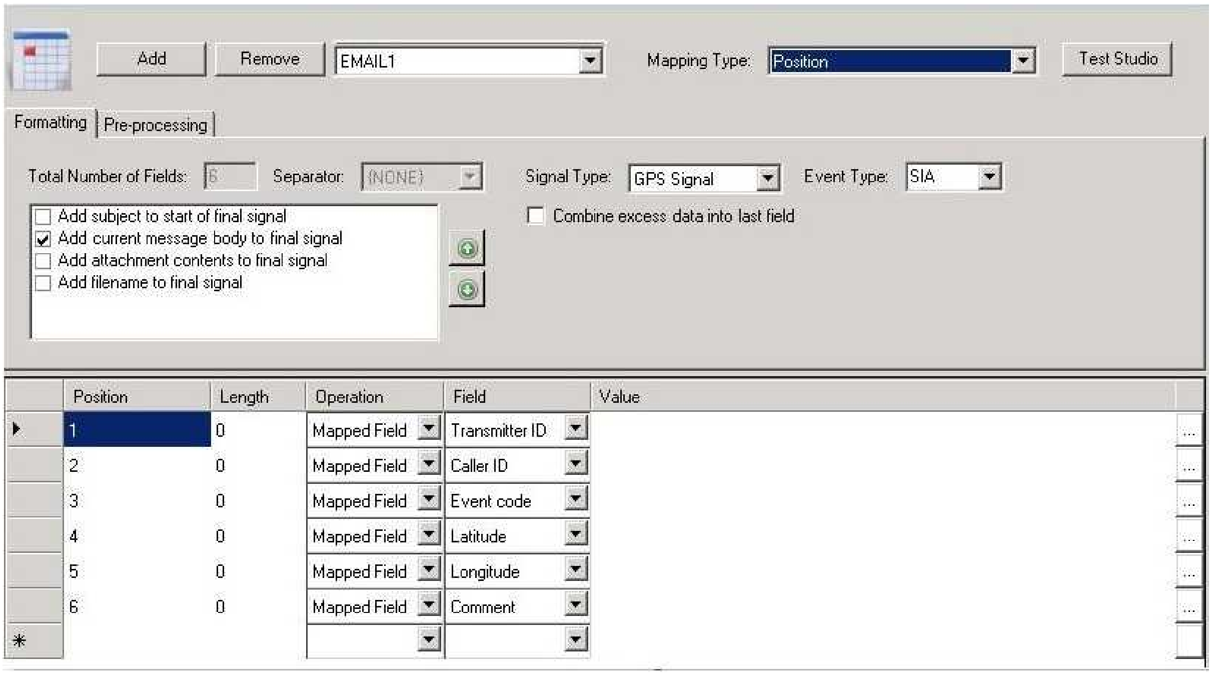
6. Select an option from the "Separator:" dropdown menu.
7. Select an option from the "Signal Type:" dropdown menu.

Note: your options are: Signal, GPS Signal, and Telemetry. For more information on these signal types, please refer to the document entitled "Field Descriptions".

8. Select an option from the "Event Type:" dropdown menu.

Note: your options are: SYS, SIA, and CID.

9. If you want to add part of the message to the final signal, select any of the four checkboxes.
10. If you want, select the "Combine excess data into the last field" checkbox.
11. In the Data Parameters section in the bottom portion of the screen, designate the order you want the information to be parsed into Manitou in the "Position" column.



Total Number of Fields: 6 Separator: (NONE) Signal Type: GPS Signal Event Type: SIA

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [ ] Combine excess data into last field

Position	Length	Operation	Field	Value
1	0	Mapped Field	Transmitter ID	
2	0	Mapped Field	Caller ID	
3	0	Mapped Field	Event code	
4	0	Mapped Field	Latitude	
5	0	Mapped Field	Longitude	
6	0	Mapped Field	Comment	

Note: the column labels in the Data Parameters area change depending on the Mapping Type you selected.

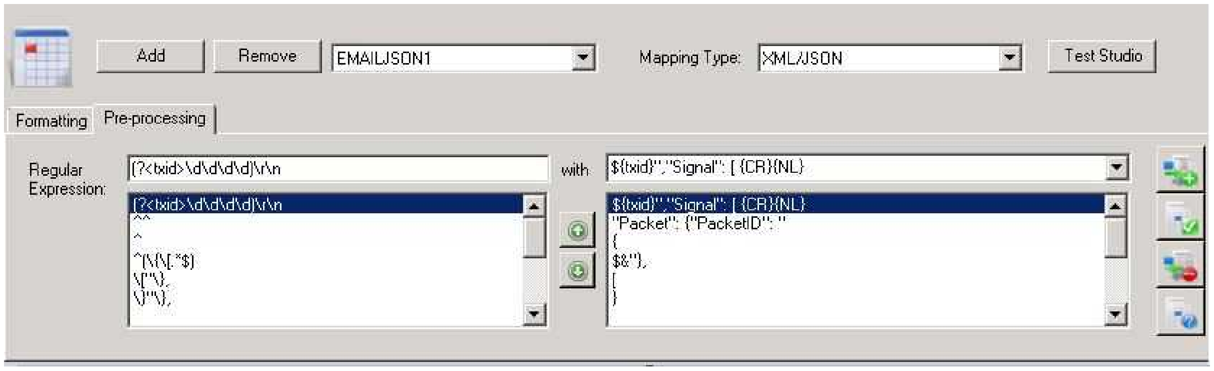
Data Mapping, Pre-processing

Defining Expressions in the Pre-processing Tab

The Pre-processing tab enables you to anticipate and prevent some common processing issues before they occur. For example, if a comma was designated as the separator on the Formatting tab, but there are instances where a semi-colon might come through the received signal as the separator, you can avoid this issue before it occurs.

Perform the following steps to use the Pre-processing tab to avoid common processing issues:

1. Click the Pre-processing tab on the Data Mapping form.
Result: the Pre-processing tab displays as shown in the following screenshot:



Add | Remove | EMAILJSON1 | Mapping Type: XML/JSON | Test Studio

Formatting | Pre-processing

Regular Expression: (?<txid>\d\d\d\d)\r\n with ${txid}","Signal": [{CR}{NL}

```
(?<txid>\d\d\d\d)\r\n
^^
^
^(\{\[.*$)
\["\},
\}"\},
```

```
${txid}","Signal": [ {CR}{NL}
"Packet": {"PacketID": "
{
$&"},
[
}
```

2. In the "Regular Expression:" field, enter the symbol or letters used as a common expression in the signals you receive.
3. Enter the interpretation option you want for the common expression in the right-hand field.

The buttons on the right-hand side of the window perform the following functions:

Add regular expression

Update regular expression

Remove regular expression

Test Studio

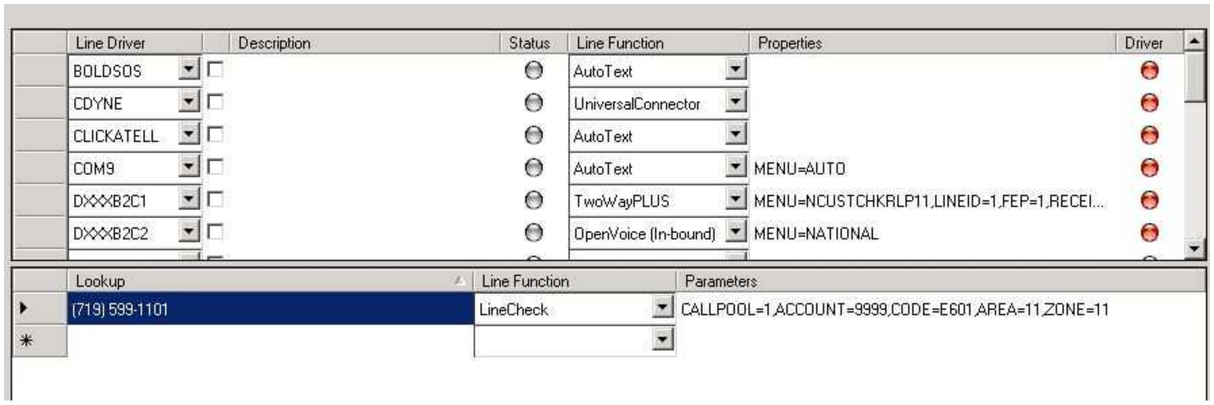
Line Drivers

Once you have created an Email Connector and mapped it, you must format the Line Drivers.

Note: the FEP and associated Receivers must be correctly installed and configured prior to formatting Line Drivers. To check the installation of the FEPs and Receivers, please refer to the appropriate Supervisor Workstation manual for your version of Manitou.

Perform the following steps to format the Line Drivers:

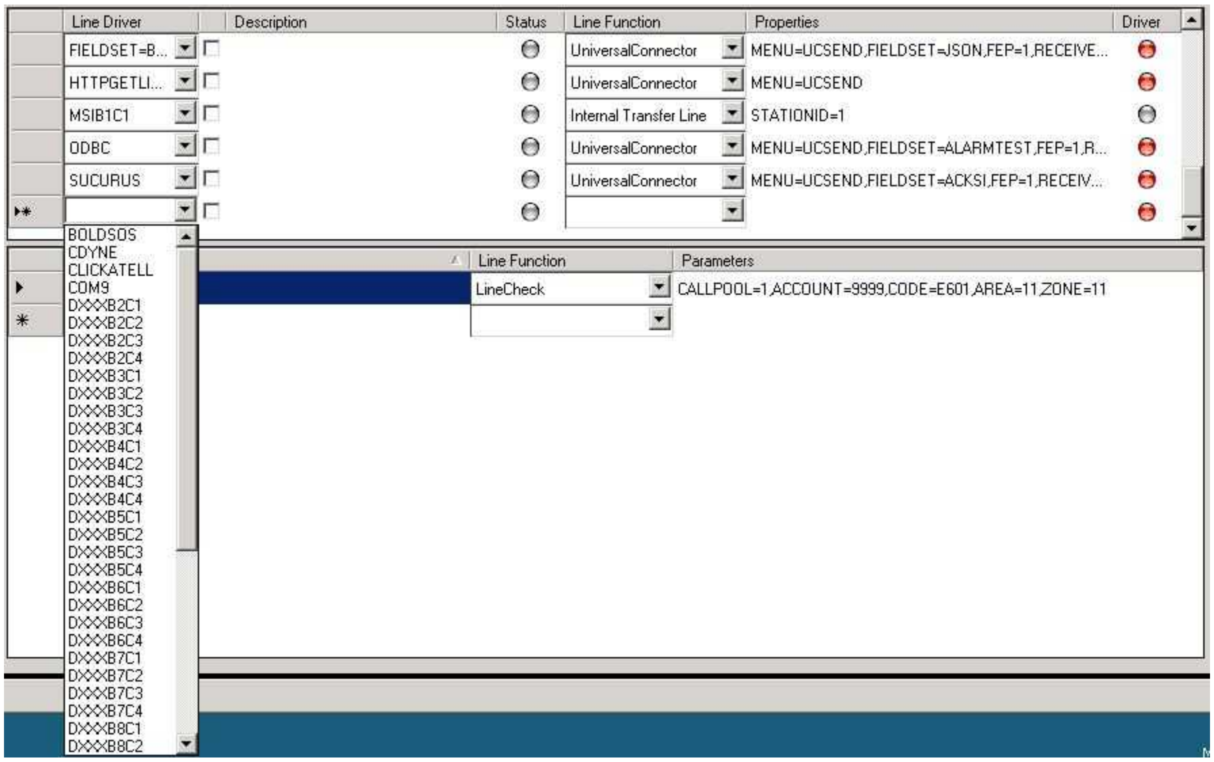
1. Click "Line Drivers" under Administration in the Node Tree.
Result: the Line Drivers form displays as shown in the following screenshot:



Line Driver		Description	Status	Line Function	Properties	Driver
BOLDSOS	☐		●	AutoText		●
CDYNE	☐		●	UniversalConnector		●
CLICKATELL	☐		●	AutoText		●
COM9	☐		●	AutoText	MENU=AUTO	●
DXXXB2C1	☐		●	TwoWayPLUS	MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	●
DXXXB2C2	☐		●	OpenVoice (In-bound)	MENU=NATIONAL	●

	Lookup	Line Function	Parameters
▶	(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*			

2. In the "Line Driver" column, select the Line Driver from the pulldown list in the next open row (indicated with an asterisk) as shown in the following screenshot:

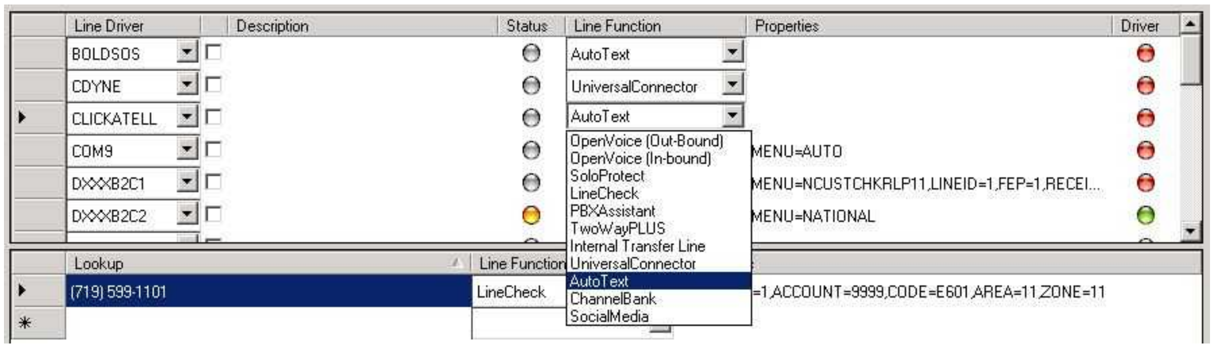


	Line Driver		Description	Status	Line Function	Properties	Driver
	FIELDSET=B...	☐		●	UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	●
	HTTPGETLI...	☐		●	UniversalConnector	MENU=UCSEND	●
	MSIB1C1	☐		●	Internal Transfer Line	STATIONID=1	●
	ODBC	☐		●	UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	●
	SUCURUS	☐		●	UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	●
▶*		☐		●			●

	Line Function	Parameters
▶	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*		

3. If you want the Line Driver to be designated as active, select the checkbox to the right of the Driver. If you do not want it to be active, leave the checkbox deselected.
4. Enter a description into the "Description" field.
5. Select the type of "Line Function" you want from the pulldown menu as shown in the

following screenshot:

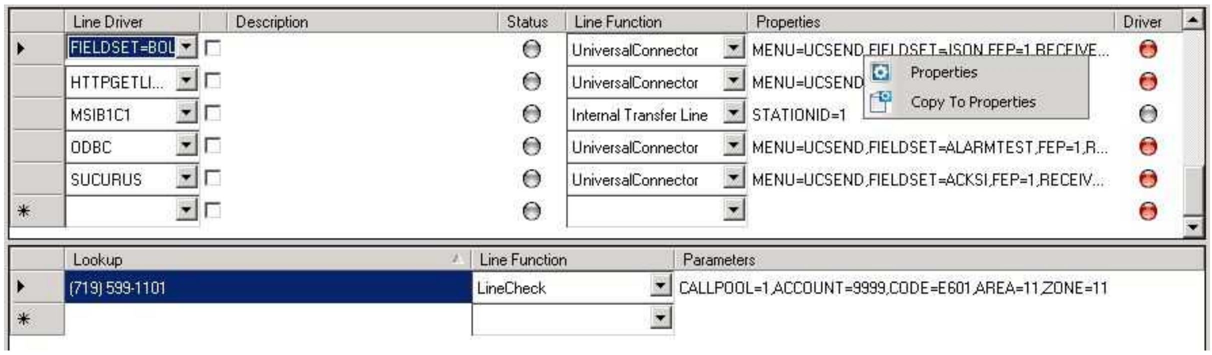


	Line Driver	Description	Status	Line Function	Properties	Driver
	BOLDSOS		●	AutoText		●
	CDYNE		●	UniversalConnector		●
▶	CLICKATELL		●	AutoText		●
	COM9		●		MENU=AUTO	●
	DXXXB2C1		●		MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	●
	DXXXB2C2		●		MENU=NATIONAL	●

	Lookup	Line Function	
▶	(719) 599-1101	LineCheck	=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*			

Establish Properties

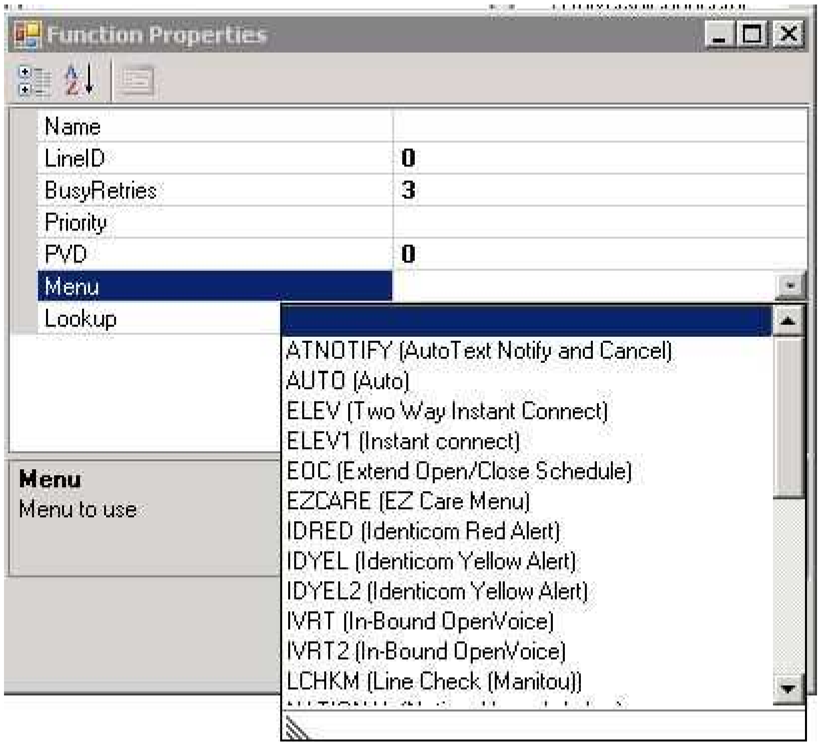
1. To enter property information for a Line Driver, select the Driver you want. Then, right-click in the "Properties" column and select Properties from the menu that displays. You also have the option of copying properties from an already existing Line Driver by selecting the "Copy to Properties" option as shown in the following screenshot:



	Line Driver	Description	Status	Line Function	Properties	Driver
▶	FIELDSET=BOL		●	UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	●
	HTTPGETLI...		●	UniversalConnector	MENU=UCSEND	●
	MSIB1C1		●	Internal Transfer Line	STATIONID=1	●
	ODBC		●	UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	●
	SUCURUS		●	UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	●
*			●			●

	Lookup	Line Function	Parameters
▶	(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*			

Result: the Function Properties window displays as shown in the following screenshot:


Function Properties

Name	
LineID	0
BusyRetries	3
Priority	
PVD	0
Menu	
Lookup	

Menu
Menu to use

ATNOTIFY (AutoText Notify and Cancel)
AUTO (Auto)
ELEV (Two Way Instant Connect)
ELEV1 (Instant connect)
EOC (Extend Open/Close Schedule)
EZCARE (EZ Care Menu)
IDRED (Identicom Red Alert)
IDYEL (Identicom Yellow Alert)
IDYEL2 (Identicom Yellow Alert)
IVRT (In-Bound OpenVoice)
IVRT2 (In-Bound OpenVoice)
LCHKM (Line Check (Manitou))

Note: the function list items that display on the Function Properties menu depend on the Line Driver you selected.

2. Enter the values you want in the right column to correspond to the function list on the left. Some functions have dropdown menus (as shown in the screenshot above). To access a dropdown menu, click at the edge of the right column until a down arrow displays. Click the arrow, and scroll through the list to select the item you want.
3. Click "**OK**" when finished, and then click "**Save**".

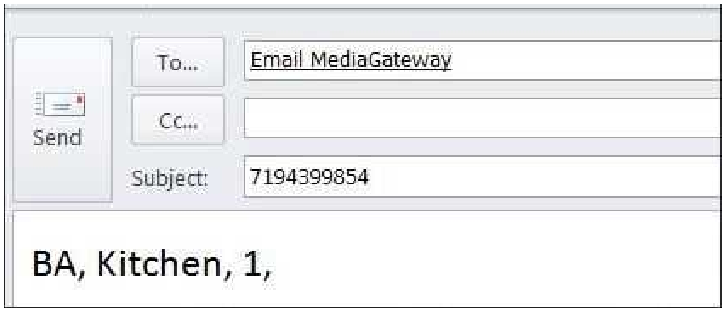
Email Example

Email Signal (Comma Separated)

Emails are received and monitored by the UniversalConnector. Once detected, they are picked up and parsed into the Manitou system. The following is an example of an email signal interpreted through the UniversalConnector.
Emails can be received in a simple comma separated format, or a more complex format that includes attachments.

Simple Signal Example

- Subject: 7194399854
- Email Body: BA, Kitchen, 1



Email MediaGateway

7194399854

BA, Kitchen, 1,

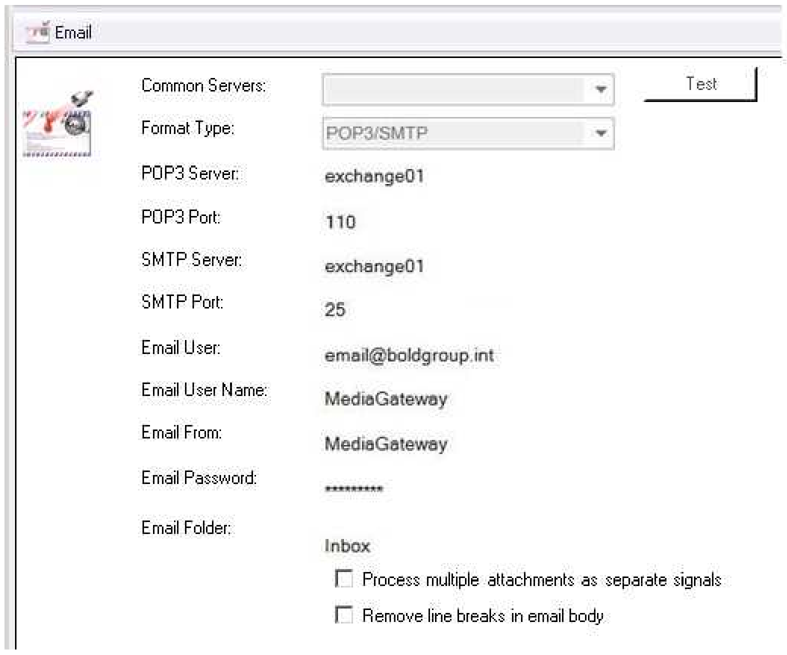
Email example

We can determine the following information from the signal received:

- BA - Event (Burglary Alarm)
- Kitchen - Point ID
- 1 - Zone

Connector

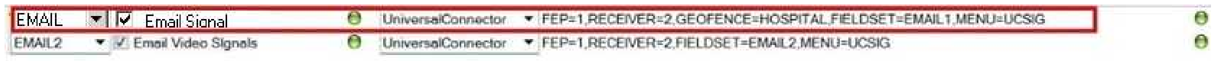
The connector configuration includes POP3 and SMTP Server and Port information for sending and receiving information from the mail server.



Common Servers:	
Format Type:	POP3/SMTP
POP3 Server:	exchange01
POP3 Port:	110
SMTP Server:	exchange01
SMTP Port:	25
Email User:	email@boldgroup.int
Email User Name:	MediaGateway
Email From:	MediaGateway
Email Password:	**********
Email Folder:	Inbox

☐ Process multiple attachments as separate signals
☐ Remove line breaks in email body

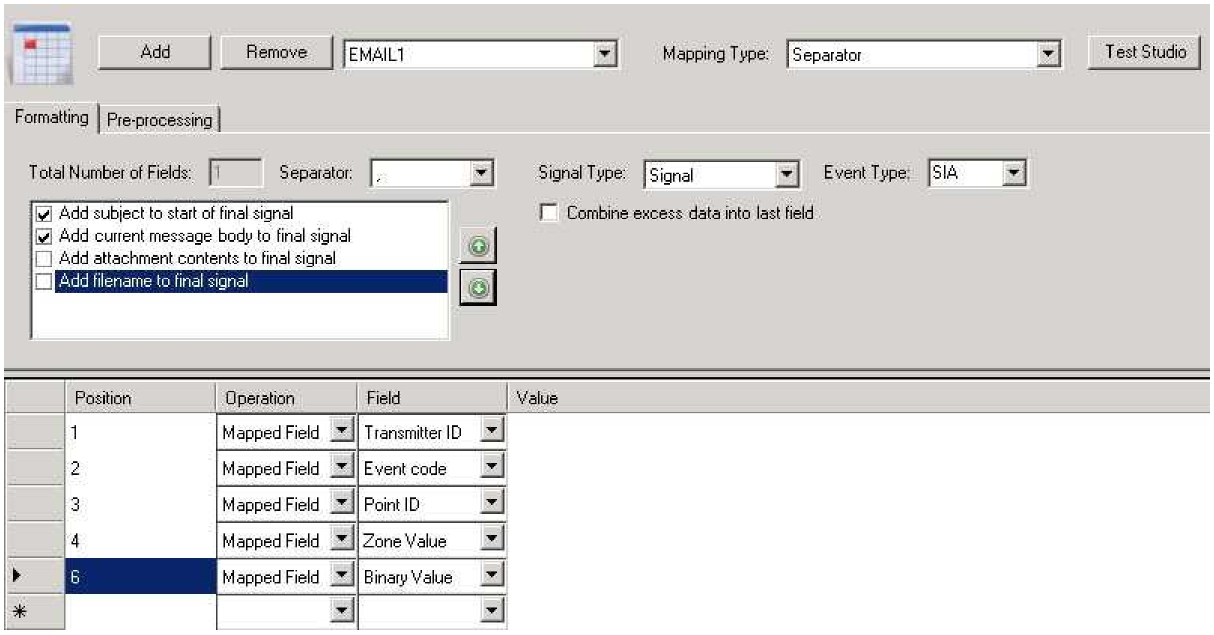
Line Drivers



EMAIL	✓	Email Signal	UniversalConnector	FEP=1,RECEIVER=2,GEOFENCE=HOSPITAL,FIELDSET=EMAIL1,MENU=UCSIG	
EMAIL2	✓	Email Video Signals	UniversalConnector	FEP=1,RECEIVER=2,FIELDSET=EMAIL2,MENU=UCSIG	

Data Mapping

Data mapping for the simple email example should match the following screenshot:


Add | Remove | EMAIL1 | Mapping Type: Separator | Test Studio

Formatting | Pre-processing

Total Number of Fields: 1 Separator: , Signal Type: Signal Event Type: SIA

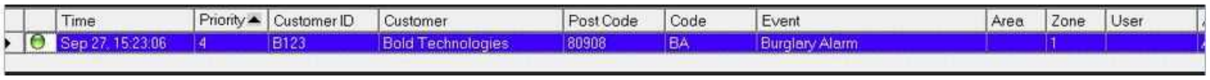
- [x] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [ ] Combine excess data into last field

Position	Operation	Field	Value
1	Mapped Field	Transmitter ID	
2	Mapped Field	Event code	
3	Mapped Field	Point ID	
4	Mapped Field	Zone Value	
6	Mapped Field	Binary Value	

Manitou Alarm

Once parsed into the system, the Manitou alarm displays as shown in the following screenshot:



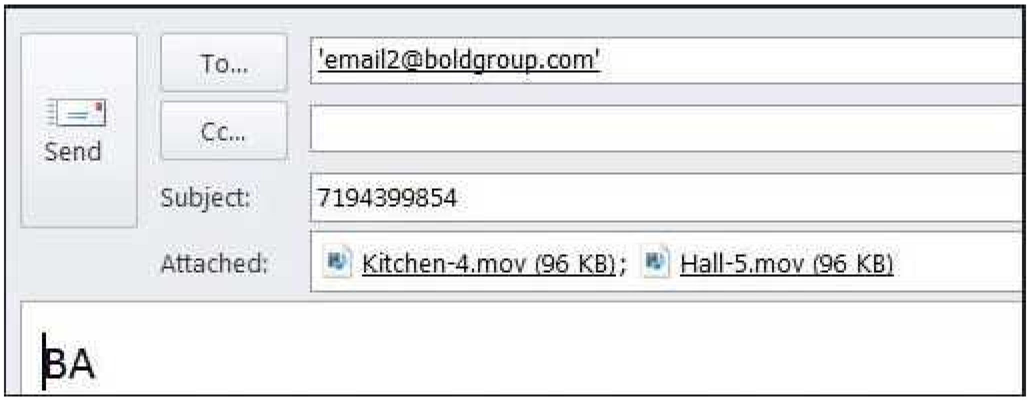
Time	Priority	Customer ID	Customer	Post Code	Code	Event	Area	Zone	User
Sep 27, 15:23:06	4	B123	Bold Technologies	80908	BA	Burglary Alarm		1	

Email Alarm example

Complex Email Signal Example

The following is an example of an email signal with video attachments:

- Subject: 7194399854
- Email Body: BA, Garage Drive, 3
- Attachment: Video file


Send

To... 'email2@boldgroup.com'

Cc...

Subject: 7194399854

Attached: Kitchen-4.mov (96 KB); Hall-5.mov (96 KB)

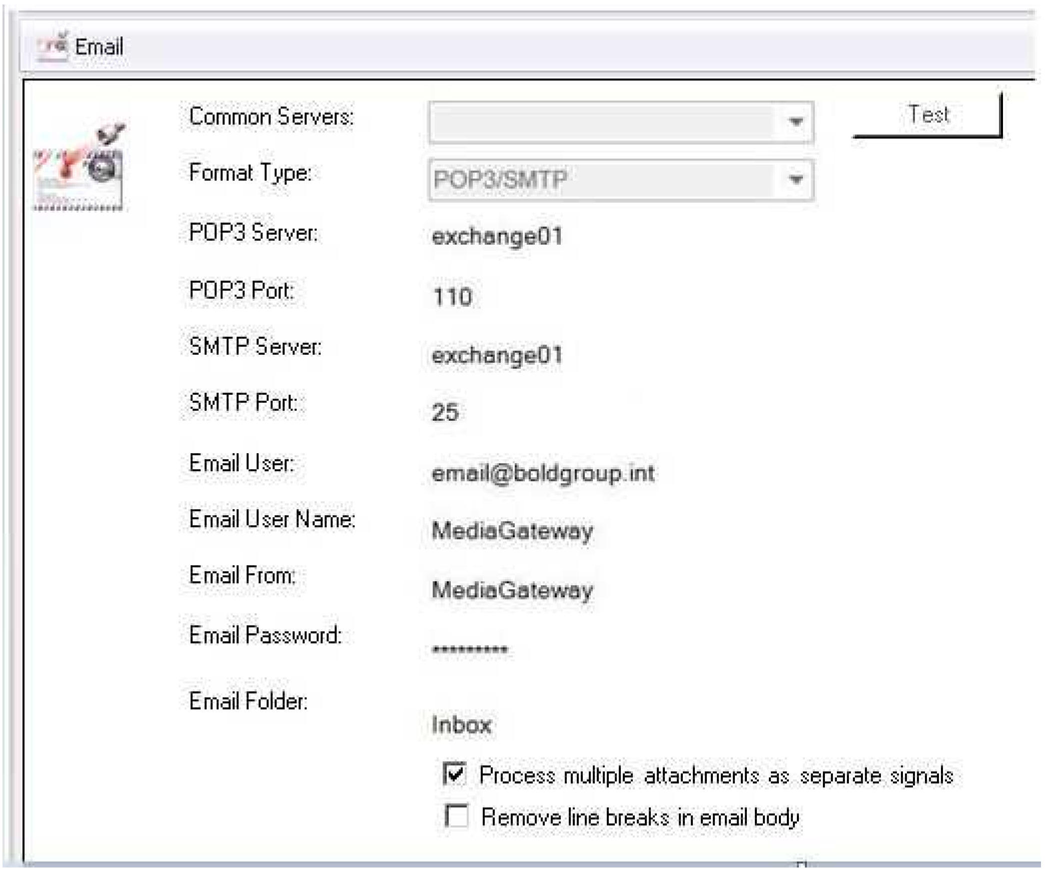
BA

We can determine the following information from the email received:

- BA - Event (Burglary Alarm)
- Garage Drive - Point ID
- 3 - Zone
- 2 accompanying video files

Connector

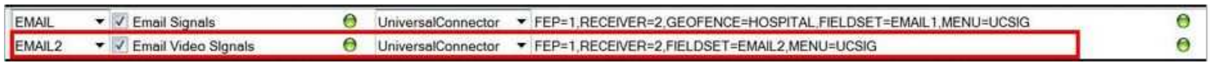
Using the above example, the Connector is configured in the following manner:



Common Servers:	
Format Type:	POP3/SMTP
POP3 Server:	exchange01
POP3 Port:	110
SMTP Server:	exchange01
SMTP Port:	25
Email User:	email@boldgroup.int
Email User Name:	MediaGateway
Email From:	MediaGateway
Email Password:	**********
Email Folder:	Inbox

☑ Process multiple attachments as separate signals
☐ Remove line breaks in email body

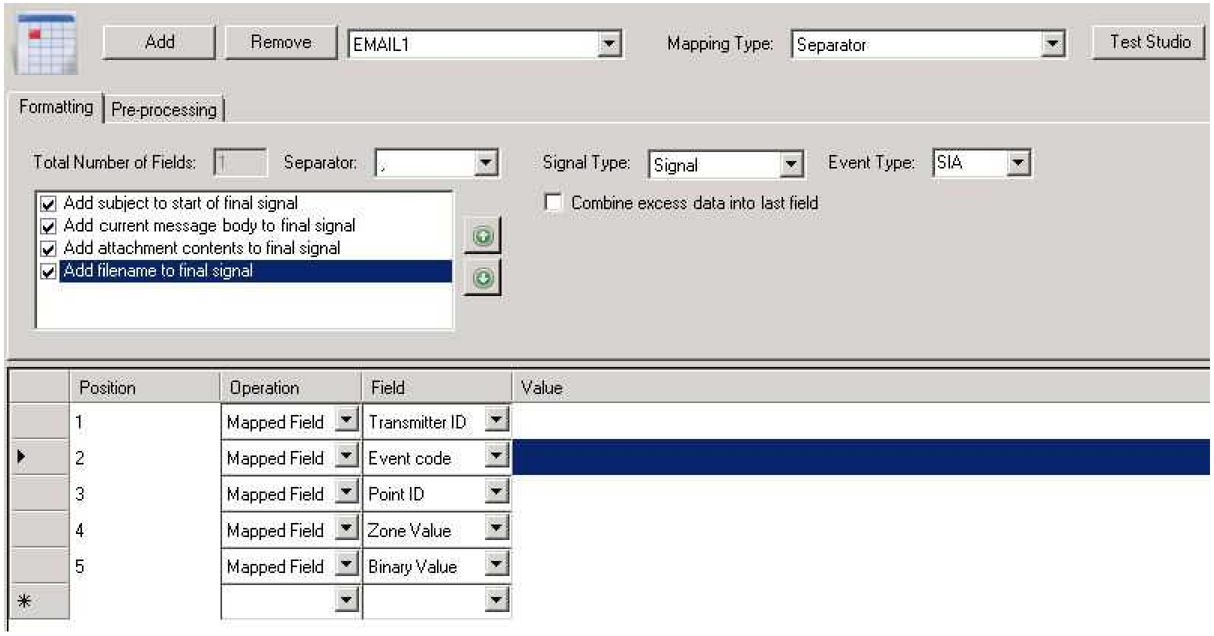
Line Drivers



EMAIL	✓ Email Signals	●	UniversalConnector	FEP=1,RECEIVER=2,GEOFENCE=HOSPITAL,FIELDSET=EMAIL1,MENU=UCSIG	●
EMAIL2	✓ Email Video Signals	●	UniversalConnector	FEP=1,RECEIVER=2,FIELDSET=EMAIL2,MENU=UCSIG	●

Email Line Drivers

Data Mapping



Add | Remove | EMAIL1 | Mapping Type: Separator | Test Studio

Formatting | Pre-processing

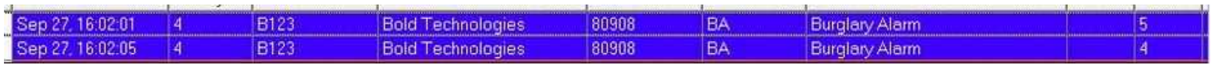
Total Number of Fields: 1 Separator: , Signal Type: Signal Event Type: SIA

- [x] Add subject to start of final signal
- [x] Add current message body to final signal
- [x] Add attachment contents to final signal
- [x] Add filename to final signal

- [ ] Combine excess data into last field

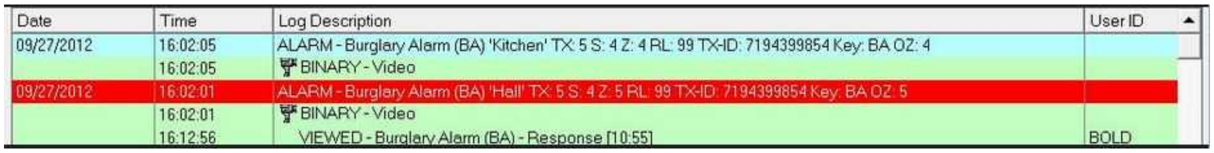
Position	Operation	Field	Value
1	Mapped Field	Transmitter ID	
2	Mapped Field	Event code	
3	Mapped Field	Point ID	
4	Mapped Field	Zone Value	
5	Mapped Field	Binary Value	

Manitou Alarm



Sep 27, 16:02:01	4	B123	Bold Technologies	80908	BA	Burglary Alarm		5
Sep 27, 16:02:05	4	B123	Bold Technologies	80908	BA	Burglary Alarm		4

Email Alarms example



Date	Time	Log Description	User ID
09/27/2012	16:02:05	ALARM - Burglary Alarm (BA) 'Kitchen' TX: 5 S: 4 Z: 4 RL: 99 TX-ID: 7194399854 Key: BA OZ: 4	
	16:02:05	BINARY - Video	
09/27/2012	16:02:01	ALARM - Burglary Alarm (BA) 'Hall' TX: 5 S: 4 Z: 5 RL: 99 TX-ID: 7194399854 Key: BA OZ: 5	
	16:02:01	BINARY - Video	
	16:12:56	VIEWED - Burglary Alarm (BA) - Response [10:55]	BOLD

Email Alarm example, Log description

The video clips display in Manitou as shown in the following screenshot:

Alarm Video Clip

ODBC

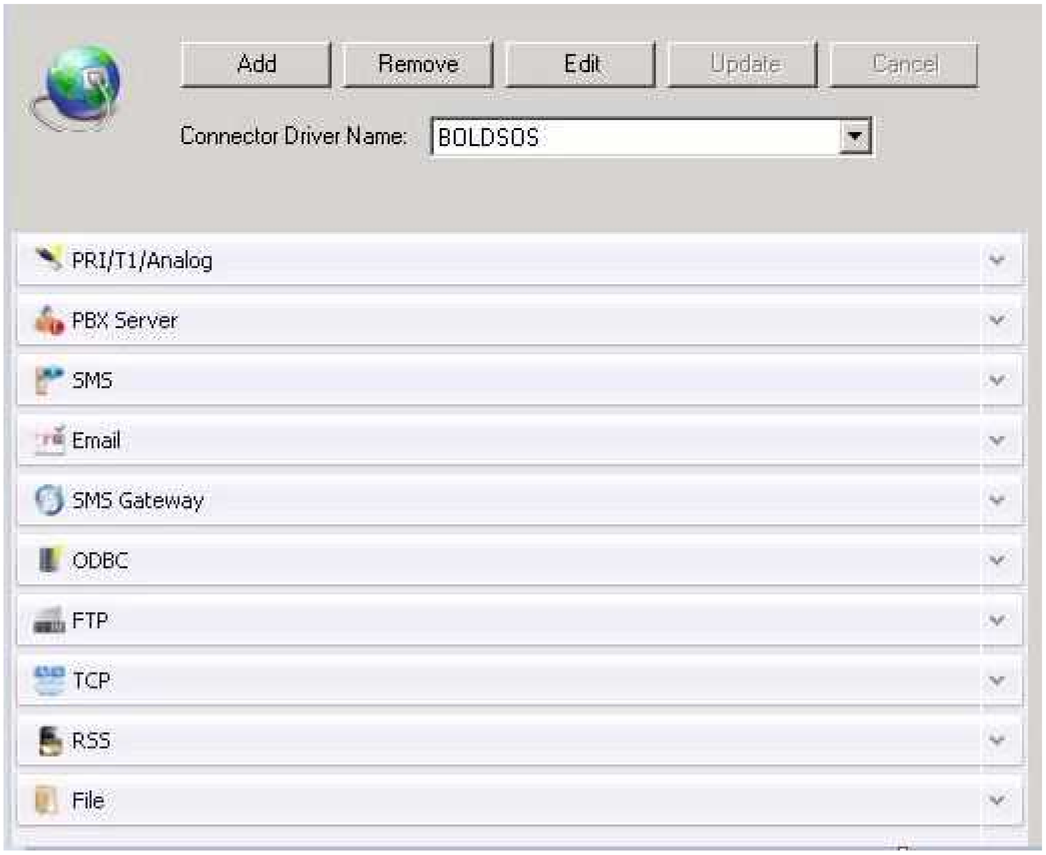
Adding an ODBC Connector

Use the ODBC Connector to connect to a database. It easily maps to specific database tables, and specific columns to mapped fields.

Perform the following steps to add an ODBC Connector:

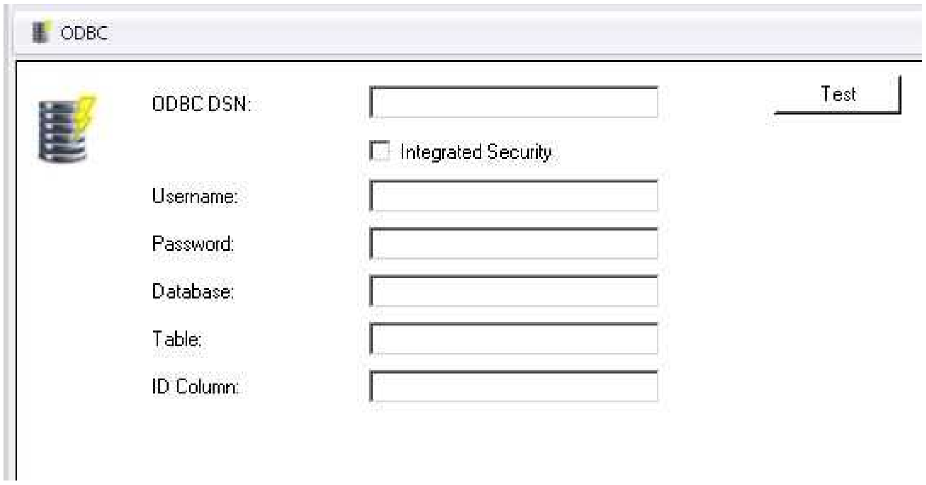
1. Click "Connector" under the UniversalConnector section of the Node Tree.

Result: the following window displays:



Connector Driver Name: BOLDSOS

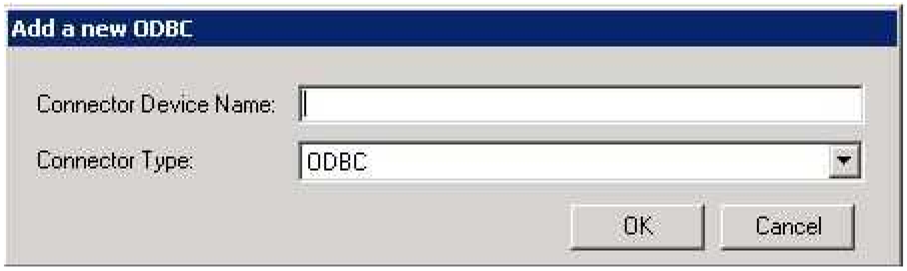
2. Click to the down arrow to expand the ODBC area of the window as shown in the following screenshot:



ODBC configuration

3. Click "**Add**".

Result: the "Add a new ODBC" dialog displays as shown in the following screenshot:



4. Enter a device name into the "Connector Device Name:" field.
5. Confirm that the "Connector Type:" field displays as "ODBC". If it does not, select "ODBC" from the dropdown list.
6. Click "**OK**".

Result: the "Add a new ODBC" window closes and the fields on the ODBC area of the window now display as populated with the default settings.

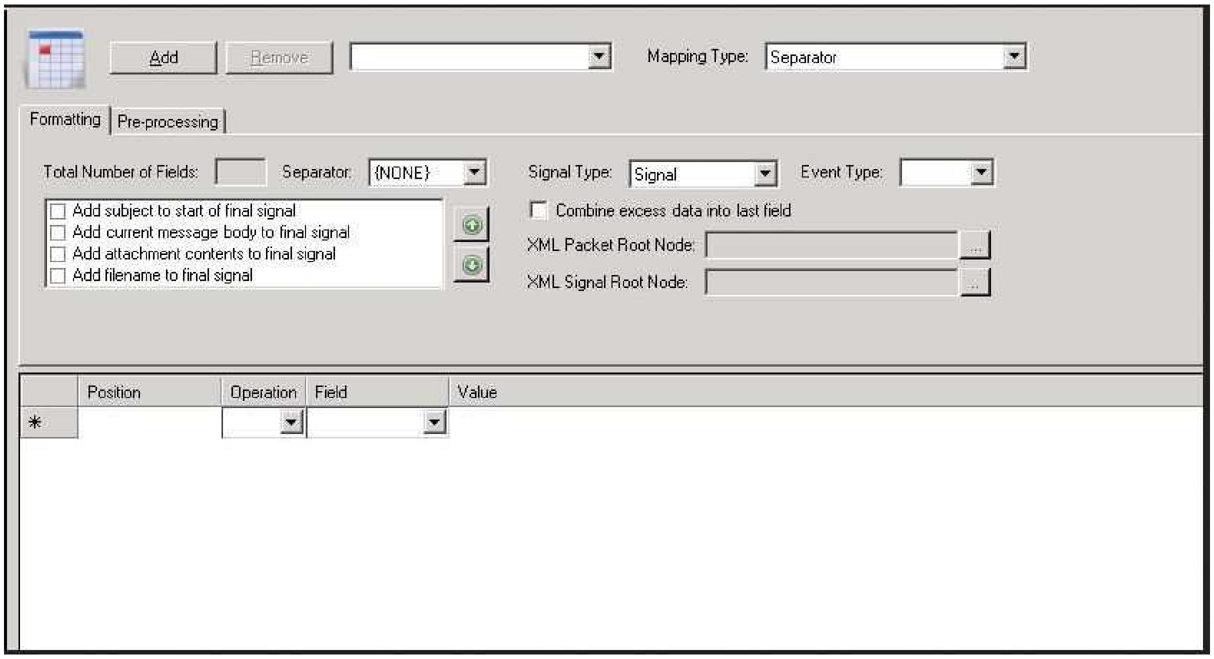
Note: you may need to modify the default settings depending on the hardware components you have installed. Use the dropdown arrows to access additional selections available for each field.

Data Mapping

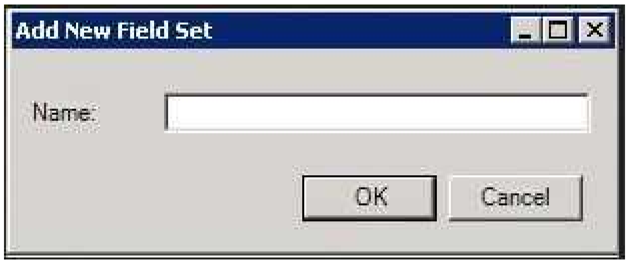
In order for the communication to be properly identified and submitted to Manitou, you must define the correct data mapping in the Data Mapping form.

Perform the following steps to define the correct data mapping:

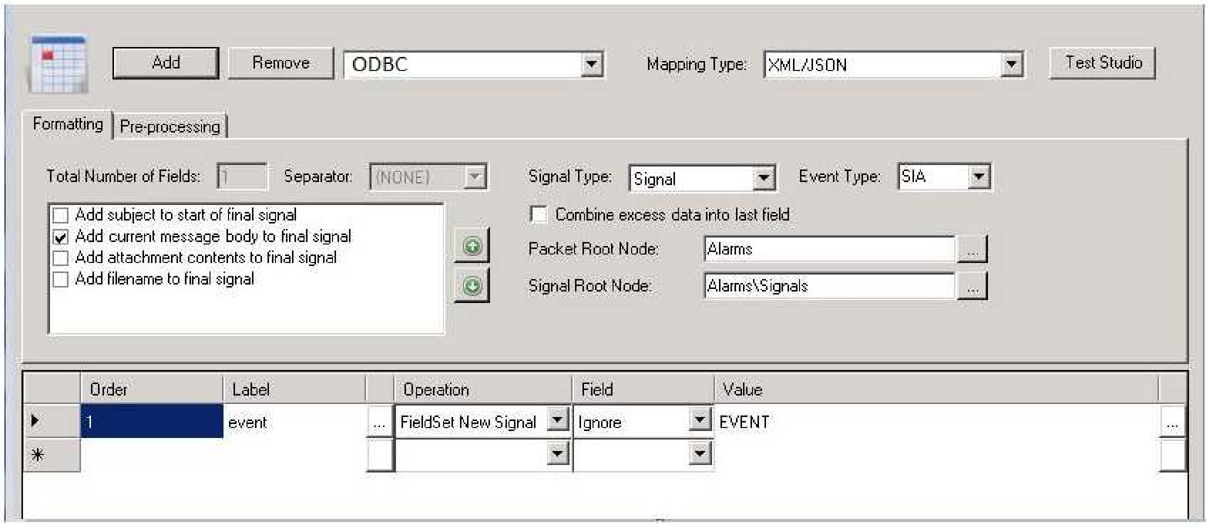
1. Click "Data Mapping" under UniversalConnector on the Node Tree.
Result: the Data Mapping form displays as shown in the following screenshot:



2. Select "ODBC" from the dropdown menu directly to the right of the "Remove" button.
3. Click "**Add**".
Result: the "Add New Field Set" dialog displays as shown in the following screenshot:



4. Enter a name into the "Name:" field, and then click "**OK**".
Result: the "Add New Field Set" window closes and the system returns you to the Data Mapping form as shown in the following screenshot:



ODBC

Mapping Type: XML/JSON

Total Number of Fields: 1 Separator: (NONE)

Signal Type: Signal Event Type: SIA

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [ ] Combine excess data into last field

Packet Root Node: Alarms

Signal Root Node: Alarms\Signals

Order	Label	Operation	Field	Value
1	event	FieldSet New Signal	Ignore	EVENT

5. Select an option from the "Mapping Type:" dropdown menu.

Note: your options are: Separator, Position, Label/Separator, and XML/JSON.

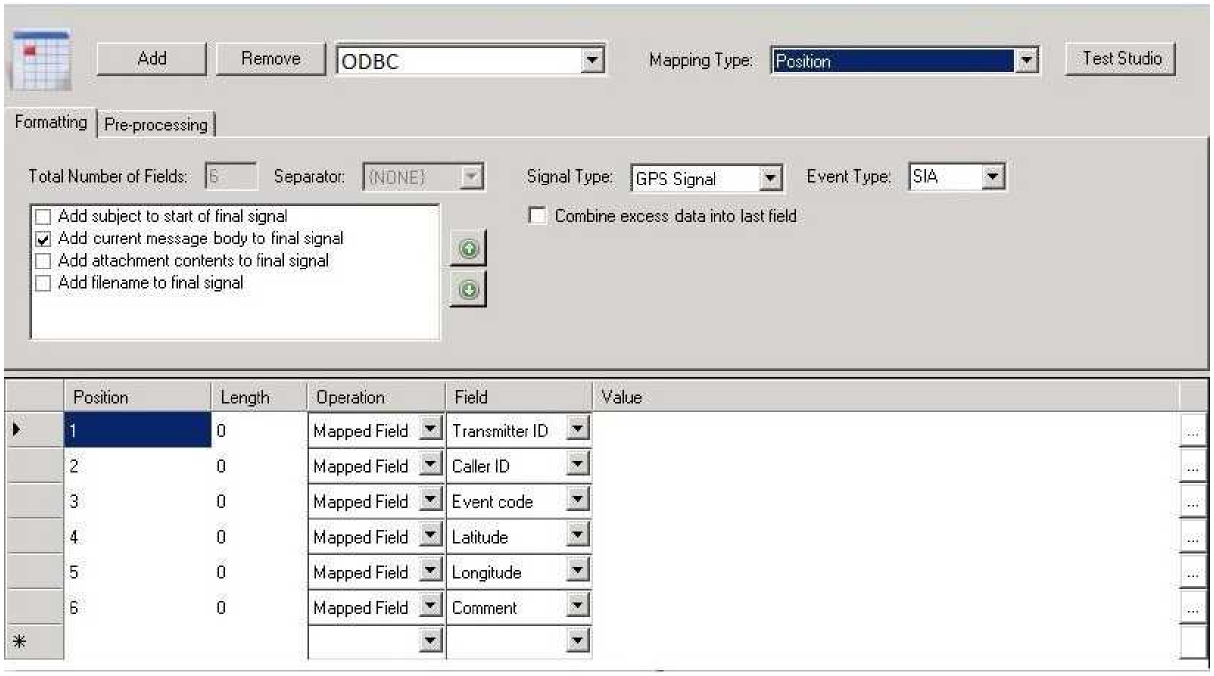
6. Select an option from the "Separator:" dropdown menu.
7. Select an option from the "Signal Type:" dropdown menu.

Note: your options are: Signal, GPS Signal, and Telemetry. For more information on these signal types, please refer to the document entitled "Field Descriptions".

8. Select an option from the "Event Type:" dropdown menu.

Note: your options are: SYS, SIA, and CID.

9. If you want to add part of the message to the final signal, select any of the four checkboxes.
10. If you want, select the "Combine excess data into the last field" checkbox.
11. In the Data Parameters section in the bottom portion of the screen, designate the order you want the information to be parsed into Manitou in the "Position" column.



ODBC

Mapping Type: Position

Total Number of Fields: 6 Separator: (NONE) Signal Type: GPS Signal Event Type: SIA

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [ ] Combine excess data into last field

Position	Length	Operation	Field	Value
1	0	Mapped Field	Transmitter ID	
2	0	Mapped Field	Caller ID	
3	0	Mapped Field	Event code	
4	0	Mapped Field	Latitude	
5	0	Mapped Field	Longitude	
6	0	Mapped Field	Comment	

Note: the column labels in the Data Parameters area change depending on the Mapping Type you selected.

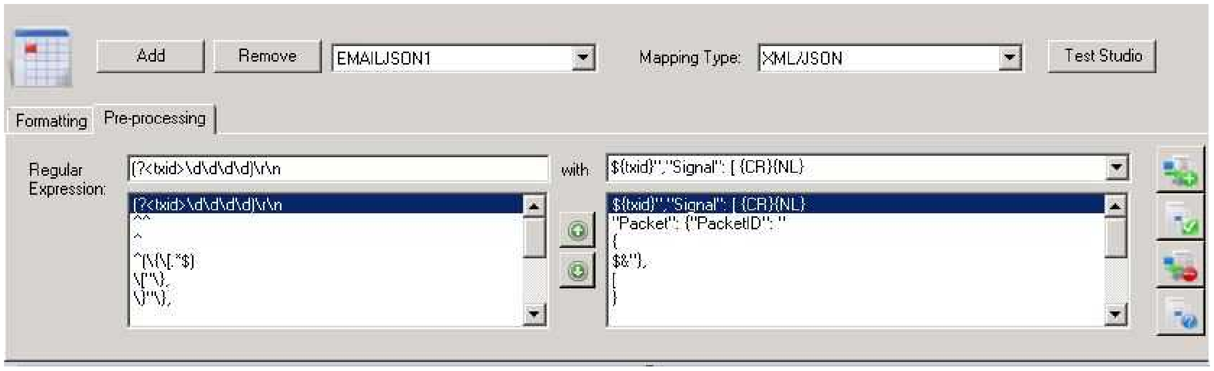
Data Mapping, Pre-processing

Defining Expressions in the Pre-processing Tab

The Pre-processing tab enables you to anticipate and prevent some common processing issues before they occur. For example, if a comma was designated as the separator on the Formatting tab, but there are instances where a semi-colon might come through the received signal as the separator, you can avoid this issue before it occurs.

Perform the following steps to use the Pre-processing tab to avoid common processing issues:

1. Click the Pre-processing tab on the Data Mapping form.
Result: the Pre-processing tab displays as shown in the following screenshot:



EMAILJSON1

Mapping Type: XML/JSON

Regular Expression: (?<txid>\d\d\d\d)\r\n with ${txid}","Signal": [{CR}{NL}

```
(?<txid>\d\d\d\d)\r\n
^^
^
^(\{\[.*$)
\["\},
\}"\},
```

```
${txid}","Signal": [ {CR}{NL}
"Packet": {"PacketID": "
{
$&"},
[
}
```

2. In the "Regular Expression:" field, enter the symbol or letters used as a common expression in the signals you receive.
3. Enter the interpretation option you want for the common expression in the right-hand field.

The buttons on the right-hand side of the window perform the following functions:

Add regular expression

Update regular expression

Remove regular expression

Test Studio

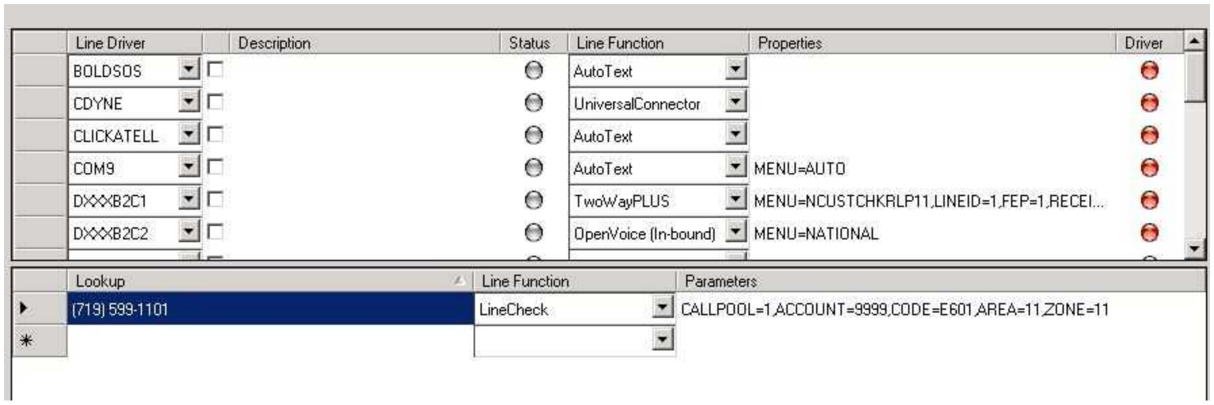
Line Drivers

Once you have created an ODBC Connector and mapped it, you must format the Line Drivers.

Note: the FEP and associated Receivers must be correctly installed and configured prior to formatting Line Drivers. To check the installation of the FEPs and Receivers, please refer to the appropriate Supervisor Workstation manual for your version of Manitou.

Perform the following steps to format the Line Drivers:

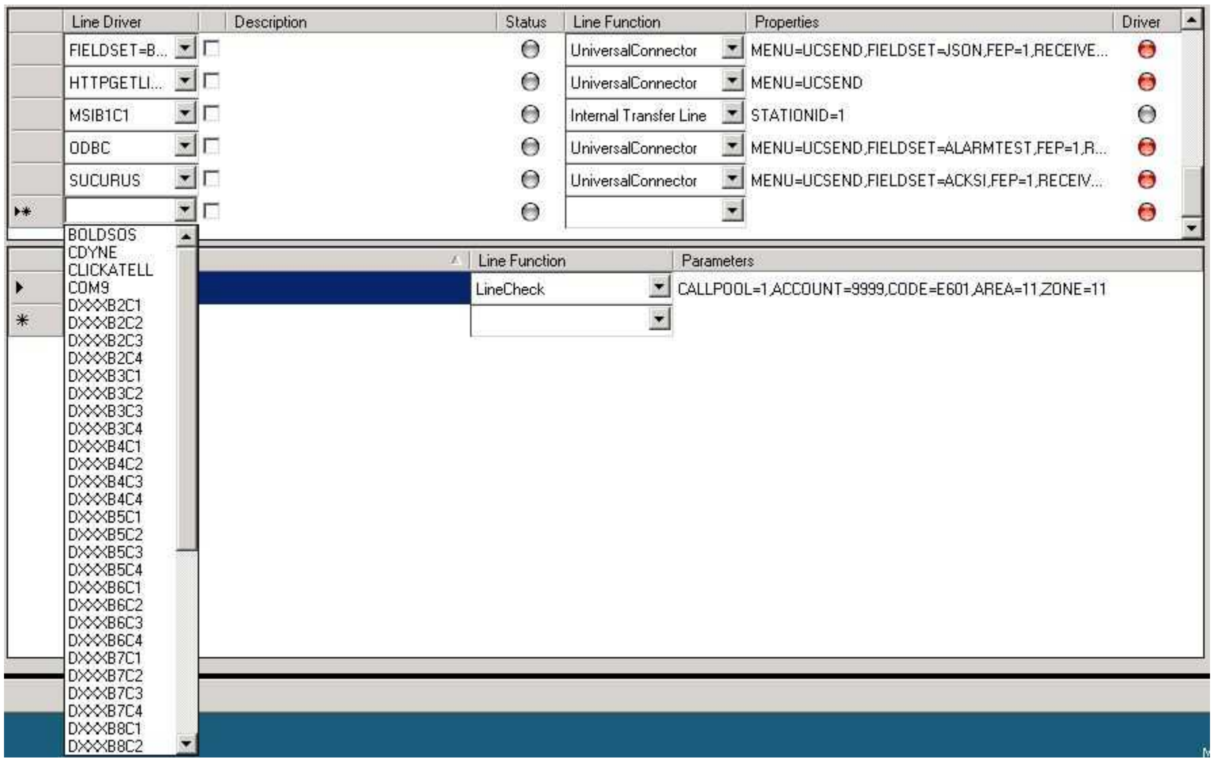
1. Click "Line Drivers" under Administration in the Node Tree.
Result: the Line Drivers form displays as shown in the following screenshot:



Line Driver		Description	Status	Line Function	Properties	Driver
BOLDSOS	☐		●	AutoText		🔴
CDYNE	☐		●	UniversalConnector		🔴
CLICKATELL	☐		●	AutoText		🔴
COM9	☐		●	AutoText	MENU=AUTO	🔴
DXXXB2C1	☐		●	TwoWayPLUS	MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	🔴
DXXXB2C2	☐		●	OpenVoice (In-bound)	MENU=NATIONAL	🔴

	Lookup	Line Function	Parameters
▶	(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*			

2. In the "Line Driver" column, select the Line Driver from the pulldown list in the next open row (indicated with an asterisk) as shown in the following screenshot:

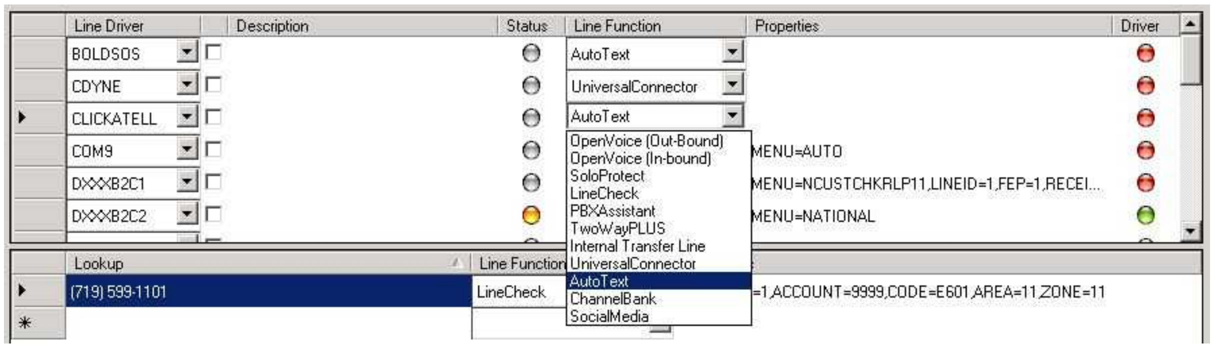


	Line Driver		Description	Status	Line Function	Properties	Driver
	FIELDSET=B...	☐		●	UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	🔴
	HTTPGETLI...	☐		●	UniversalConnector	MENU=UCSEND	🔴
	MSIB1C1	☐		●	Internal Transfer Line	STATIONID=1	●
	ODBC	☐		●	UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	🔴
	SUCURUS	☐		●	UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	🔴
▶*		☐		●			🔴

		Line Function	Parameters
▶		LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*			

3. If you want the Line Driver to be designated as active, select the checkbox to the right of it. If you do not want it to be active, leave the checkbox deselected.
4. Enter a description into the "Description" field.
5. Select the type of "Line Function" you want from the pulldown menu as shown in the

following screenshot:

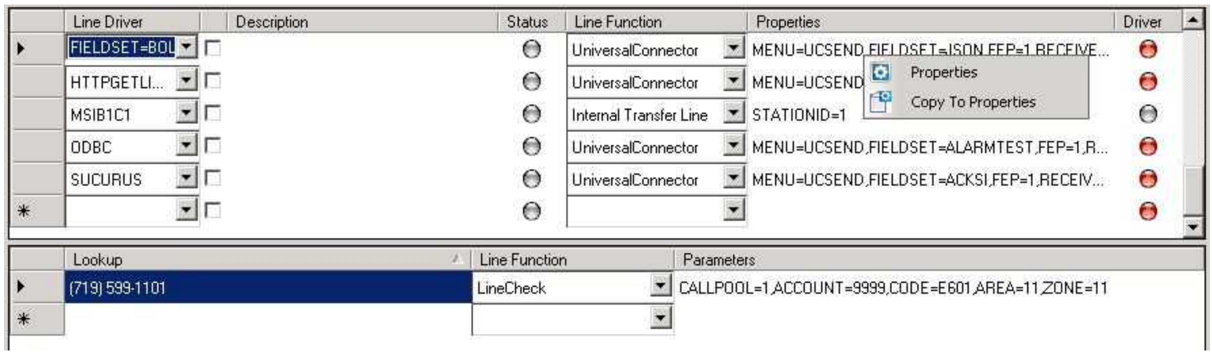


Line Driver	Description	Status	Line Function	Properties	Driver
BOLDSOS		●	AutoText		🔴
CDYNE		●	UniversalConnector		🔴
CLICKATELL		●	AutoText		🔴
COM9		●		MENU=AUTO	🔴
DXXXB2C1		●		MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	🔴
DXXXB2C2		🟠		MENU=NATIONAL	🟢

Lookup	Line Function	
(719) 599-1101	LineCheck	=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

Establish Properties

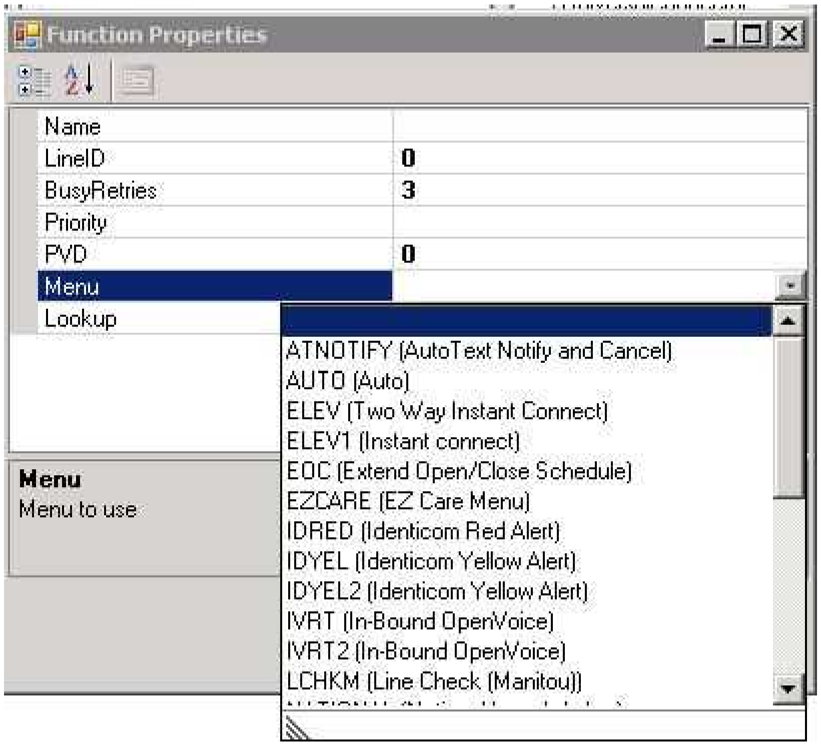
1. To enter property information for a Line Driver, select the Driver you want. Then, right-click in the "Properties" column and select Properties from the menu that displays. You also have the option of copying properties from an already existing Line Driver by selecting the "Copy to Properties" option as shown in the following screenshot:



Line Driver	Description	Status	Line Function	Properties	Driver
FIELDSET=BOL		●	UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	🔴
HTTPGETLI...		●	UniversalConnector	MENU=UCSEND	🔴
MSIB1C1		●	Internal Transfer Line	STATIONID=1	●
ODBC		●	UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	🔴
SUCURUS		●	UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	🔴
		●			🔴

Lookup	Line Function	Parameters
(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

Result: the Function Properties window displays as shown in the following screenshot:



Name	
LineID	0
BusyRetries	3
Priority	
PVD	0
Menu	
Lookup	

Menu
Menu to use

ATNOTIFY (AutoText Notify and Cancel)
AUTO (Auto)
ELEV (Two Way Instant Connect)
ELEV1 (Instant connect)
EOC (Extend Open/Close Schedule)
EZCARE (EZ Care Menu)
IDRED (Identicom Red Alert)
IDYEL (Identicom Yellow Alert)
IDYEL2 (Identicom Yellow Alert)
IVRT (In-Bound OpenVoice)
IVRT2 (In-Bound OpenVoice)
LCHKM (Line Check (Manitou))

Note: the function list items that display on the Function Properties menu depend on the Line Driver you selected.

2. Enter the values you want in the right column to correspond to the function list on the left. Some functions have dropdown menus (as shown in the screenshot above). To access a dropdown menu, click at the edge of the right column until a down arrow displays. Click the arrow, and scroll through the list to select the item you want.
3. Click "**OK**" when finished, and then click "**Save**".

ODBC Example

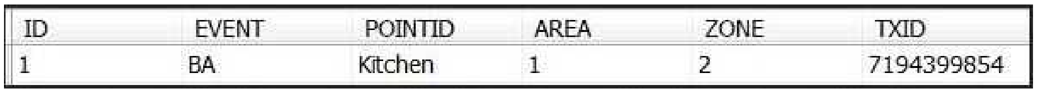
ODBC Signal

The UniversalConnector can be directed to monitor a specific database for activity to generate an alarm in Manitou.

Note: Manitou treats the ODBC format like Comma Separated. Therefore, the hierarchy of column order is important.

ODBC Signal Example

The following is an example of an ODBC signal:



ID	EVENT	POINTID	AREA	ZONE	TXID
1	BA	Kitchen	1	2	7194399854

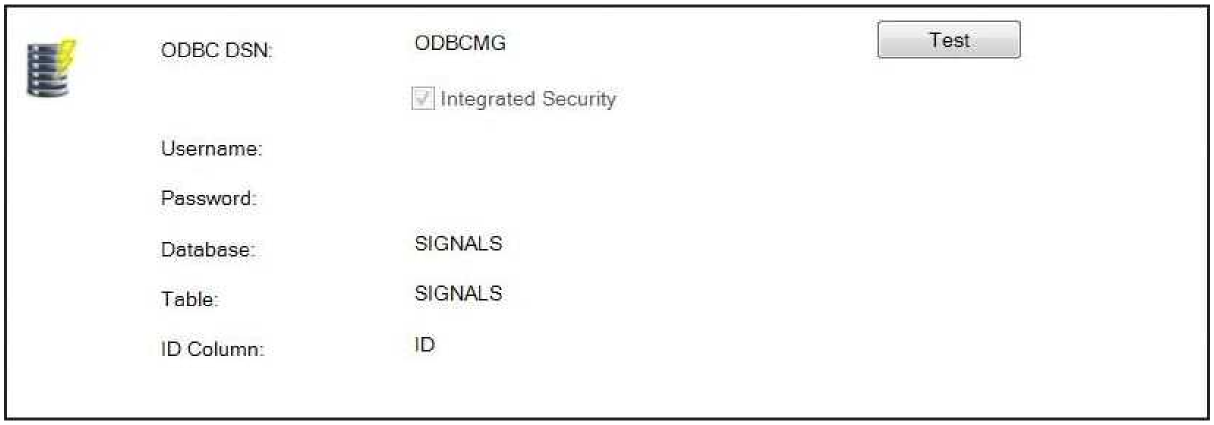
ODBC Database Value example

We can determine the following information from the signal:

- BA - Event
- Kitchen - Point ID
- 1 - Area
- 2 - Zone
- 7194399854 - TXID

Connector

The ODBC Connector configuration should match the following screenshot:

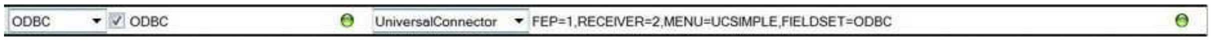


ODBC DSN:	ODBCMG
	☑ Integrated Security
Username:	
Password:	
Database:	SIGNALS
Table:	SIGNALS
ID Column:	ID

ODBC Connector example

Line Map

The ODBC line mapping should match the following screenshot:



ODBC ☑ ODBC UniversalConnector FEP=1,RECEIVER=2,MENU=UCSIMPLE,FIELDSET=ODBC

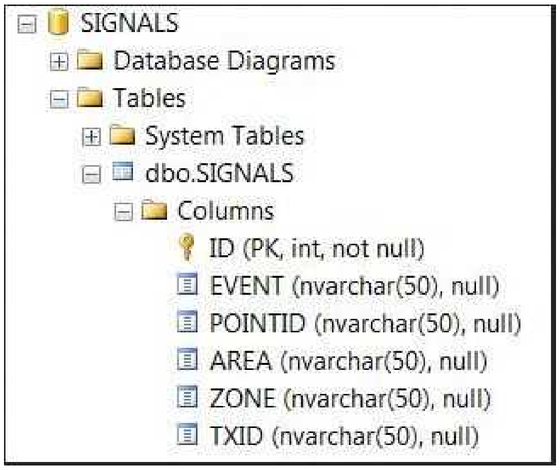
ODBC Line Mapping example

Notice that the Fieldset has been identified.

Data Mapping

In this example, the line map points to Fieldset “FEPBINARY”.

Once configured, the UniversalConnector can monitor a database table for the specified criteria.

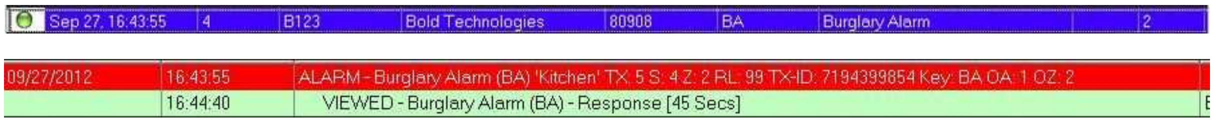


```
SIGNALS
  Database Diagrams
  Tables
    System Tables
    dbo.SIGNALS
      Columns
        ID (PK, int, not null)
        EVENT (nvarchar(50), null)
        POINTID (nvarchar(50), null)
        AREA (nvarchar(50), null)
        ZONE (nvarchar(50), null)
        TXID (nvarchar(50), null)
```

ODBC Database Table example

Manitou Alarm

Once the information in the signal example presents to the database, Manitou picks it up and generates an alarm as shown in the following screenshot:



```
Sep 27, 16:43:55 | 4 | B123 | Bold Technologies | 80908 | BA | Burglary Alarm | | 2

09/27/2012 | 16:43:55 | ALARM - Burglary Alarm (BA) 'Kitchen' TX: 5 S: 4 Z: 2 RL: 99 TX-ID: 7194399854 Key: BA OA: 1 OZ: 2
           | 16:44:40 |    VIEWED - Burglary Alarm (BA) - Response [45 Secs]                              | E
```

ODBC Generated Alarm examples

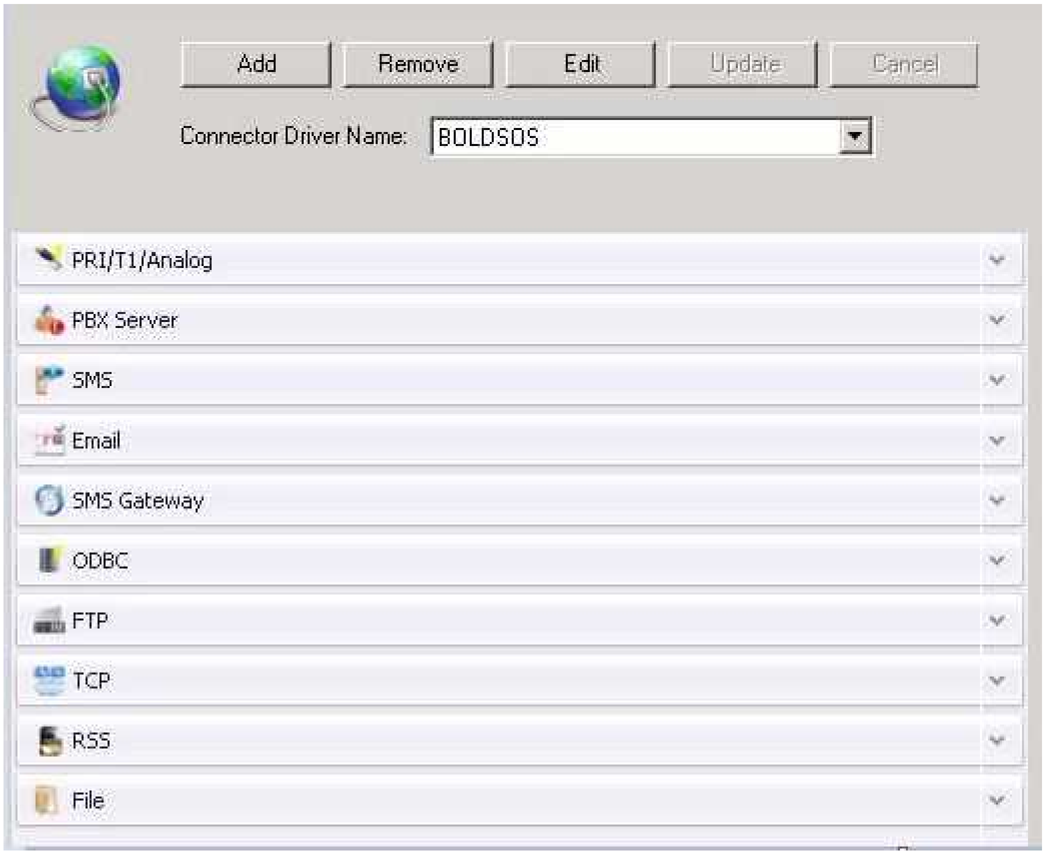
Adding an FTP Connector

The FTP Connector supports information sent over an FTP. The FTP Connector can be configured to include the following items:

- File name
- Body Contents
- Binary information such as video, audio, pictures
- Text that provides signal detail information

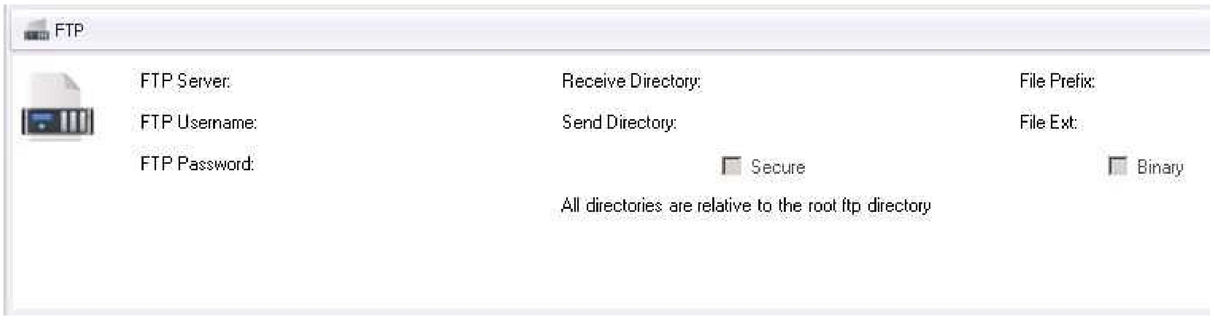
Perform the following steps to add an FTP Connector:

1. Click "Connector" under the UniversalConnector section of the Node Tree.
Result: the following window displays:



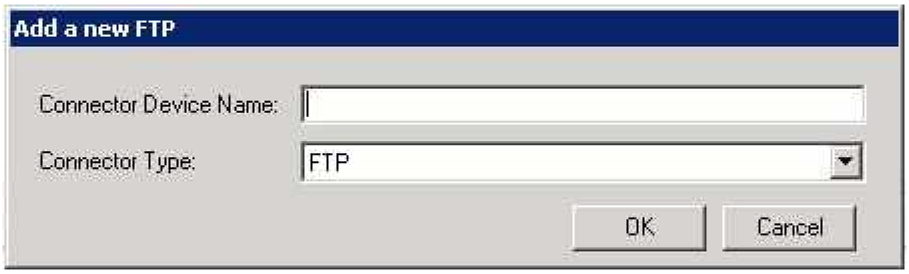
Connector Driver Name: BOLDSOS

2. Click the down arrow to expand the FTP area of the window as shown in the following screenshot:



All directories are relative to the root ftp directory

3. Click "**Add**".
Result: the "Add a new FTP" dialog displays as shown in the following screenshot:



Connector Device Name:

Connector Type: FTP

4. Enter a device name into the "Connector Device Name:" field.
5. Confirm that the "Connector Type:" field displays as "FTP". If it does not, select "FTP" from the dropdown list.
6. Click "**OK**".

Result: the "Add a new FTP" window closes and the fields on the FTP area of the window now display as populated with the default settings.

Note: you may need to modify the default settings depending on the hardware components you have installed. Use the dropdown arrows to access additional selections available for each field.

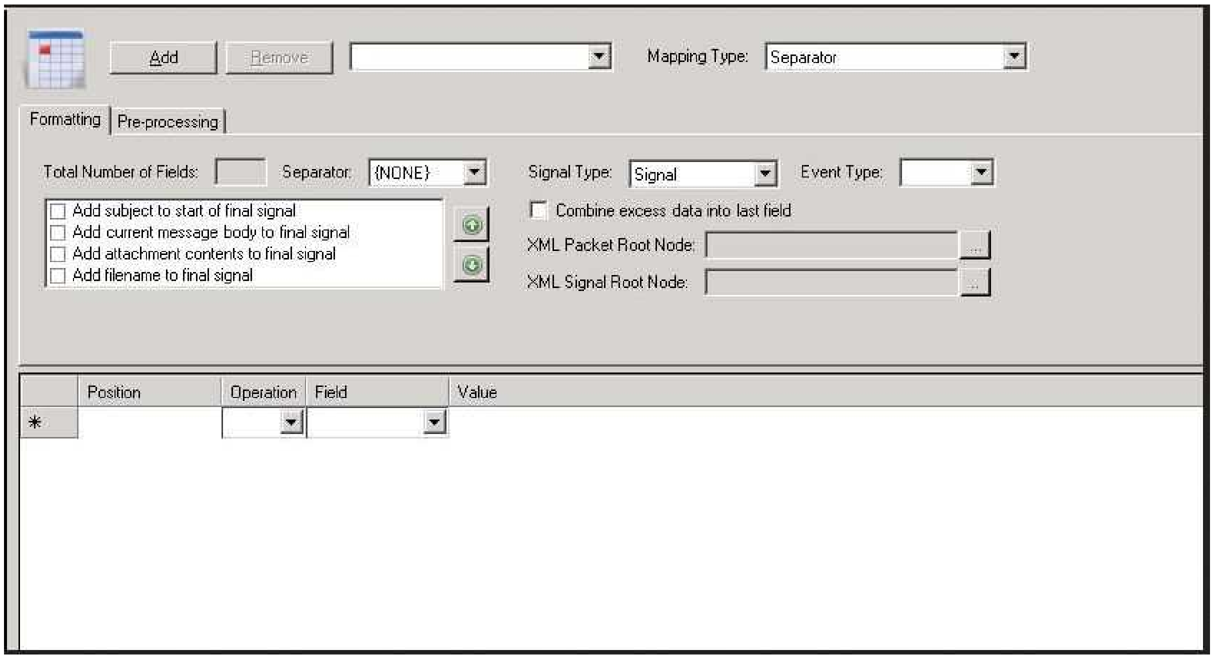
Data Mapping

In order for the communication to be properly identified and submitted to Manitou, you must define the correct data mapping in the Data Mapping form.

Perform the following steps to define the correct data mapping:

1. Click "Data Mapping" under UniversalConnector on the Node Tree.

Result: the Data Mapping form displays as shown in the following screenshot:



Mapping Type: Separator

Total Number of Fields: Separator: {NONE} Signal Type: Signal Event Type:

☐ Add subject to start of final signal
☐ Add current message body to final signal
☐ Add attachment contents to final signal
☐ Add filename to final signal

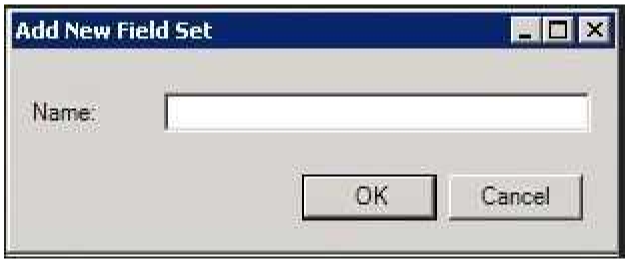
☐ Combine excess data into last field

XML Packet Root Node:

XML Signal Root Node:

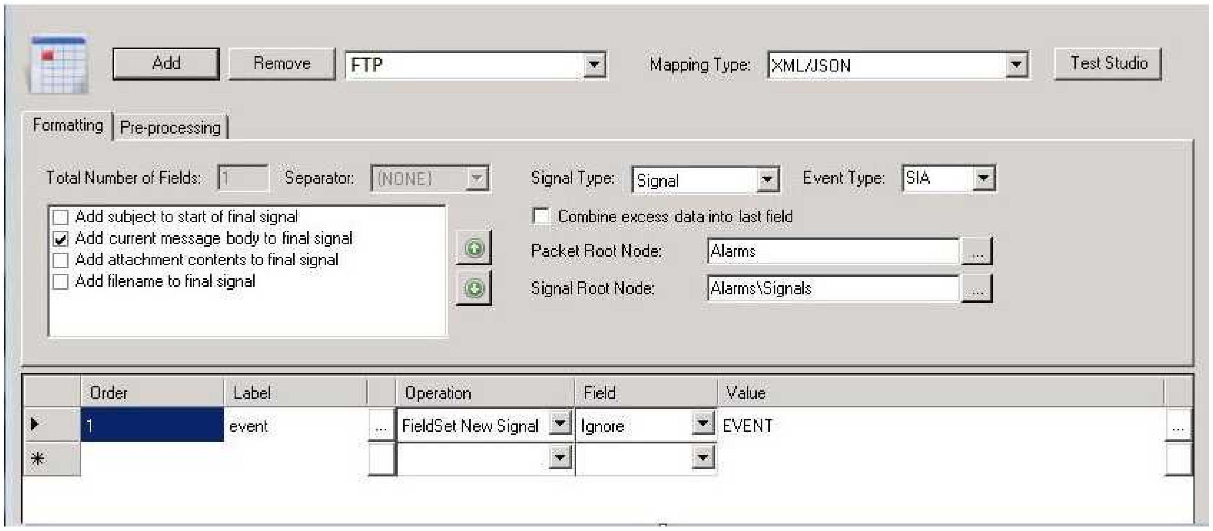
2. Select "FTP" from the dropdown menu directly to the right of the Remove button.
3. Click "**Add**".

Result: the "Add New Field Set" dialog displays as shown in the following screenshot:



4. Enter a name into the "Name:" field, and then click "**OK**".

Result: the "Add New Field Set" window closes and the system returns you to the Data Mapping form as shown in the following screenshot:



FTP

Mapping Type: XML/JSON

Total Number of Fields: 1 Separator: (NONE)

Signal Type: Signal Event Type: SIA

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [ ] Combine excess data into last field

Packet Root Node: Alarms

Signal Root Node: Alarms\Signals

Order	Label	Operation	Field	Value
1	event	FieldSet New Signal	Ignore	EVENT

5. Select an option from the "Mapping Type:" dropdown menu.

Note: your options are: Separator, Position, Label/Separator, and XML/JSON.

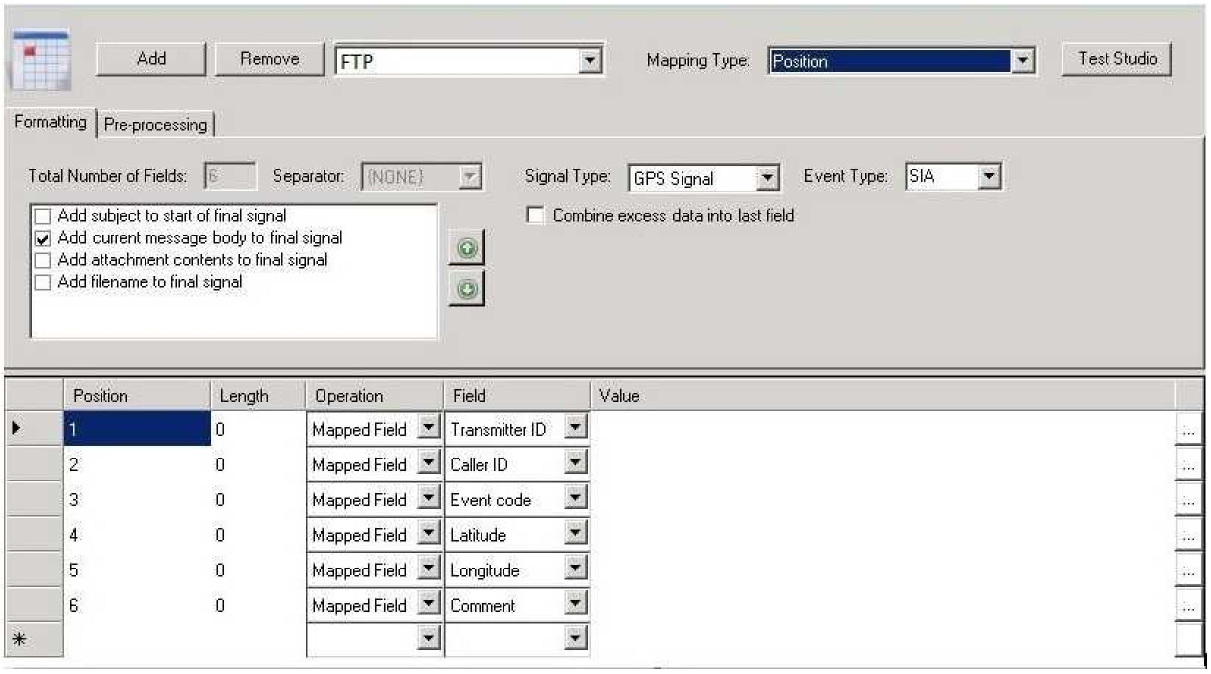
6. Select an option from the "Separator:" dropdown menu.
7. Select an option from the "Signal Type:" dropdown menu.

Note: your options are: Signal, GPS Signal, and Telemetry. For more information on these signal types, please refer to the document entitled "Field Descriptions".

8. Select an option from the "Event Type:" dropdown menu.

Note: your options are: SYS, SIA, and CID.

9. If you want to add part of the message to the final signal, select any of the four checkboxes.
10. If you want, select the "Combine excess data into the last field" checkbox.
11. In the Data Parameters section in the bottom portion of the screen, designate the order you want the information to be parsed into Manitou in the "Position" column.



FTP

Mapping Type: Position

Total Number of Fields: 6 Separator: (NONE) Signal Type: GPS Signal Event Type: SIA

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal
- [ ] Combine excess data into last field

Position	Length	Operation	Field	Value
1	0	Mapped Field	Transmitter ID	
2	0	Mapped Field	Caller ID	
3	0	Mapped Field	Event code	
4	0	Mapped Field	Latitude	
5	0	Mapped Field	Longitude	
6	0	Mapped Field	Comment	

Note: the column labels in the Data Parameters area change depending on the Mapping Type you selected.

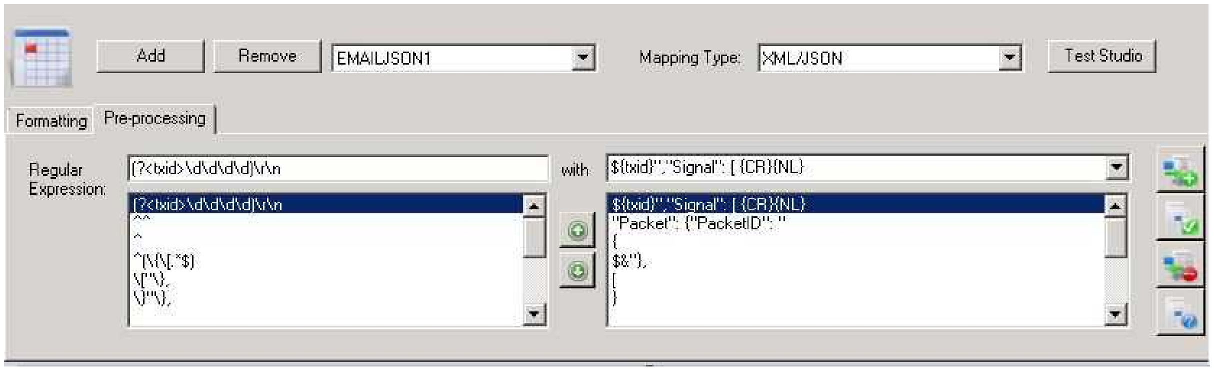
Data Mapping, Pre-processing

Defining Expressions in the Pre-processing Tab

The Pre-processing tab enables you to anticipate and prevent some common processing issues before they occur. For example, if a comma was designated as the separator on the Formatting tab, but there are instances where a semi-colon might come through the received signal as the separator, you can avoid this issue before it occurs.

Perform the following steps to use the Pre-processing tab to avoid common processing issues:

1. Click the Pre-processing tab on the Data Mapping form.
Result: the Pre-processing tab displays as shown in the following screenshot:



EMAILJSON1

Mapping Type: XML/JSON

Regular Expression: (?<txid>\d\d\d\d)\r\n with ${txid}","Signal": [{CR}{NL}

```
(?<txid>\d\d\d\d)\r\n
^^
^
^(\{\[.*$)
\["\},
\}"\},
```

```
${txid}","Signal": [{CR}{NL}
"Packet": {"PacketID": "
{
$&"},
[
}
```

2. In the "Regular Expression:" field, enter the symbol or letters used as a common expression in the signals you receive.
3. Enter the interpretation option you want for the common expression in the right-hand field.

The buttons on the right-hand side of the window perform the following functions:

Add regular expression

Update regular expression

Remove regular expression

Test Studio

Line Drivers

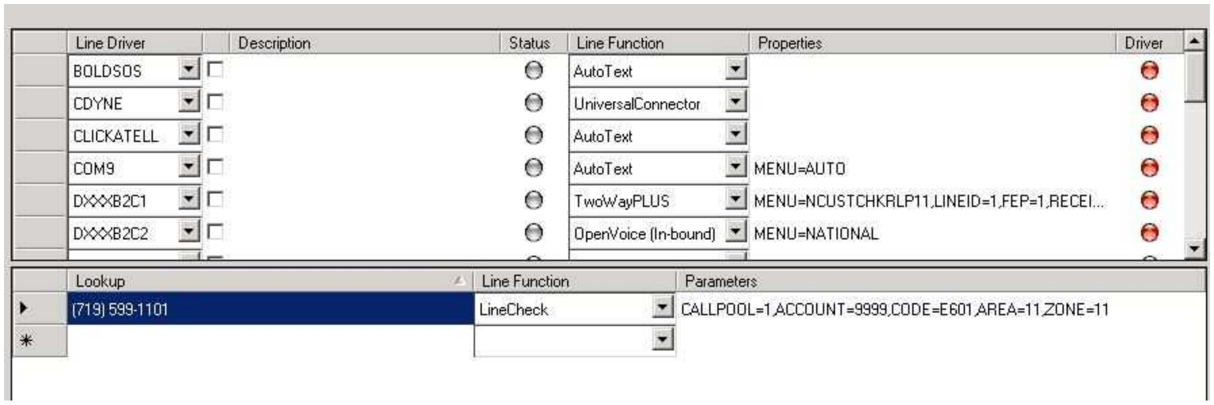
Once you have created an FTP Connector and mapped it, you must format the Line Drivers.

Note: the FEP and associated Receivers must be correctly installed and configured prior to formatting Line Drivers. To check the installation of the FEPs and Receivers, please refer to the appropriate Supervisor Workstation manual for your version of Manitou.

Perform the following steps to format the Line Drivers:

1. Click "Line Drivers" under Administration in the Node Tree.

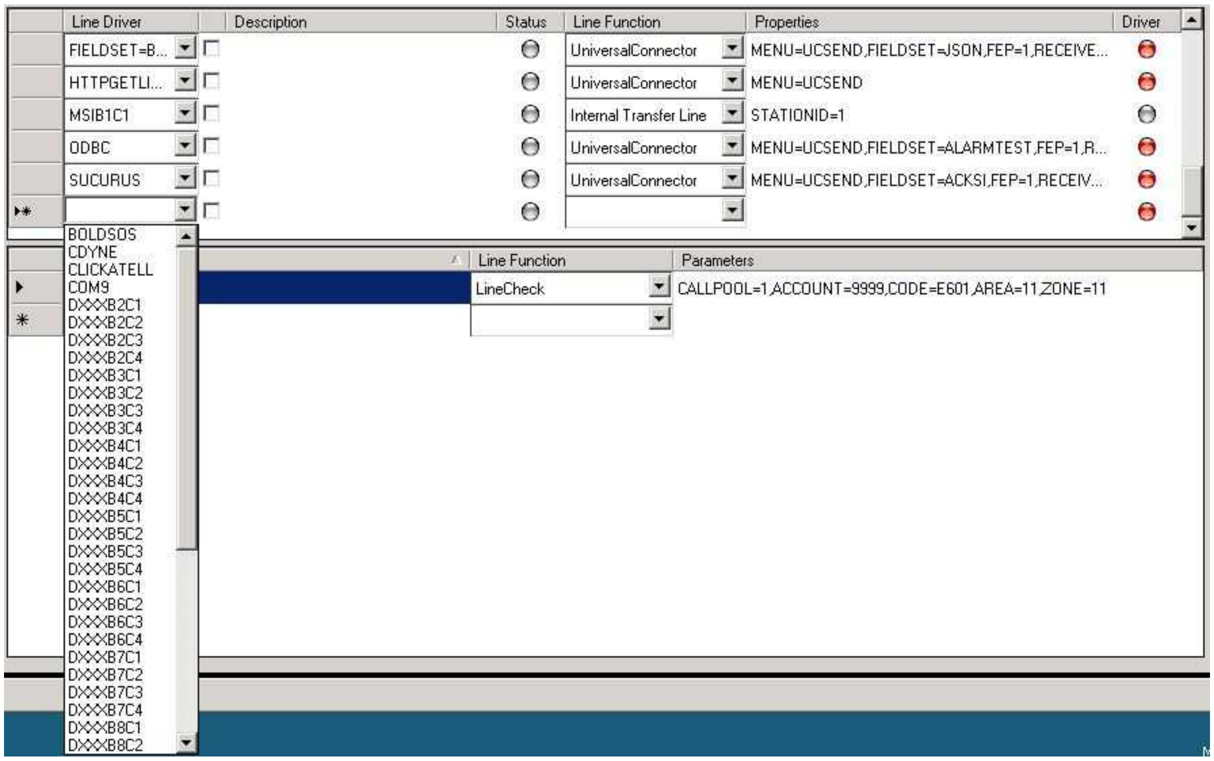
Result: the Line Drivers form displays shown in the following screenshot:



Line Driver		Description	Status	Line Function	Properties	Driver
BOLDSOS	☐		⚪	AutoText		🔴
CDYNE	☐		⚪	UniversalConnector		🔴
CLICKATELL	☐		⚪	AutoText		🔴
COM9	☐		⚪	AutoText	MENU=AUTO	🔴
DXXXB2C1	☐		⚪	TwoWayPLUS	MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	🔴
DXXXB2C2	☐		⚪	OpenVoice (In-bound)	MENU=NATIONAL	🔴

	Lookup	Line Function	Parameters
▸	(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*			

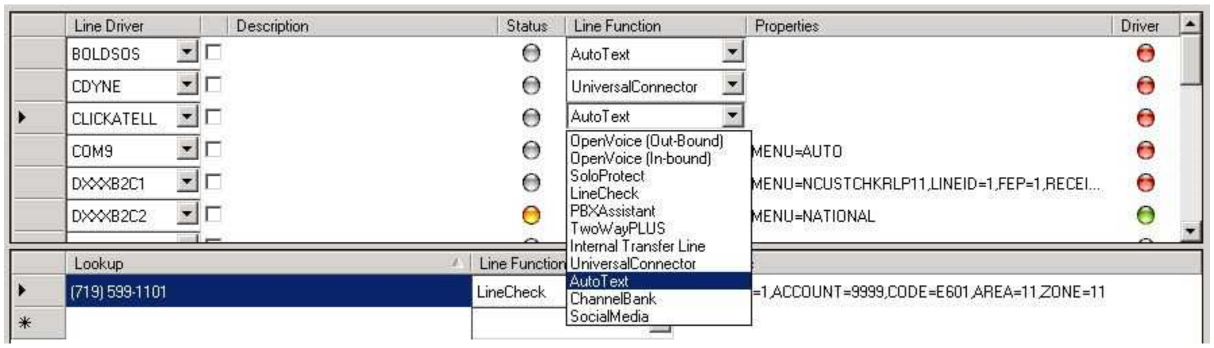
2. In the "Line Driver" column, select the Line Driver from the pulldown list in the next open row (indicated with an asterisk) as shown in the following screenshot:



	Line Driver		Description	Status	Line Function	Properties	Driver
	FIELDSET=B...	☐		⚪	UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	🔴
	HTTPGETLI...	☐		⚪	UniversalConnector	MENU=UCSEND	🔴
	MSIB1C1	☐		⚪	Internal Transfer Line	STATIONID=1	⚪
	ODBC	☐		⚪	UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	🔴
	SUCURUS	☐		⚪	UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	🔴
▸*		☐		⚪			🔴

	Line Function	Parameters
▸	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*		

3. If you want the Line Driver to be designated as active, select the checkbox to the right of the Driver. If you do not want it to be active, leave the checkbox deselected.
4. Enter a description into the "Description" field.
5. Select the type of "Line Function" you want from the pulldown menu as shown in the following screenshot:

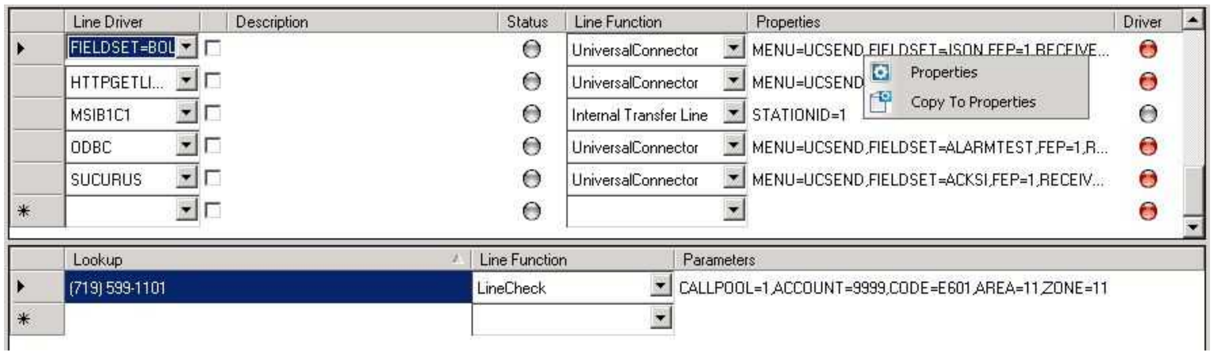


Line Driver	Description	Status	Line Function	Properties	Driver
BOLDSOS			AutoText		
CDYNE			UniversalConnector		
CLICKATELL			AutoText		
COM9				MENU=AUTO	
DXXXB2C1				MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	
DXXXB2C2				MENU=NATIONAL	

Lookup	Line Function	
(719) 599-1101	LineCheck	=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

Establish Properties

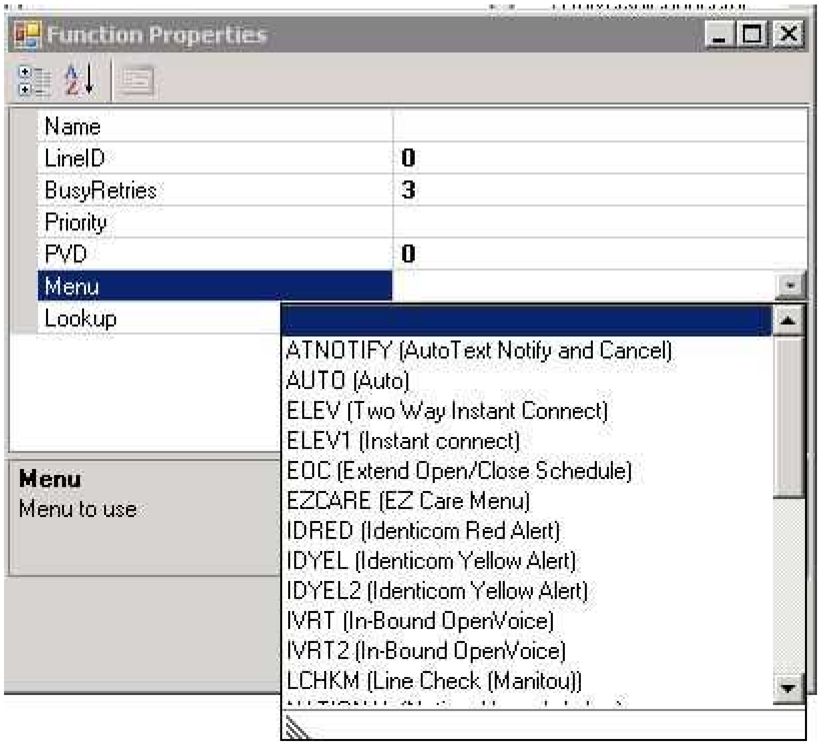
1. To enter property information for a Line Driver, select the Driver you want. Then, right-click in the "Properties" column and select Properties from the menu that displays. You also have the option of copying properties from an already existing Line Driver by selecting the "Copy to Properties" option as shown in the following screenshot:



Line Driver	Description	Status	Line Function	Properties	Driver
FIELDSET=BOL			UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	
HTTPGETLI...			UniversalConnector	MENU=UCSEND	
MSIB1C1			Internal Transfer Line	STATIONID=1	
ODBC			UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	
SUCURUS			UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	

Lookup	Line Function	Parameters
(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

Result: the Function Properties window displays as shown in the following screenshot:


Function Properties
Name
LineID 0
BusyRetries 3
Priority
PVD 0
Menu
Lookup
Menu
Menu to use
ATNOTIFY (AutoText Notify and Cancel)
AUTO (Auto)
ELEV (Two Way Instant Connect)
ELEV1 (Instant connect)
EOC (Extend Open/Close Schedule)
EZCARE (EZ Care Menu)
IDRED (Identicom Red Alert)
IDYEL (Identicom Yellow Alert)
IDYEL2 (Identicom Yellow Alert)
IVRT (In-Bound OpenVoice)
IVRT2 (In-Bound OpenVoice)
LCHKM (Line Check (Manitou))

Note: the function list items that display on the Function Properties menu depend on the Line Driver you selected.

2. Enter the values you want in the right column to correspond to the function list on the left. Some functions have dropdown menus (as shown in the screenshot above). To access a dropdown menu, click at the edge of the right column until a down arrow displays. Click the arrow, and scroll through the list to select the item you want.
3. Click "**OK**" when finished, and then click "**Save**".

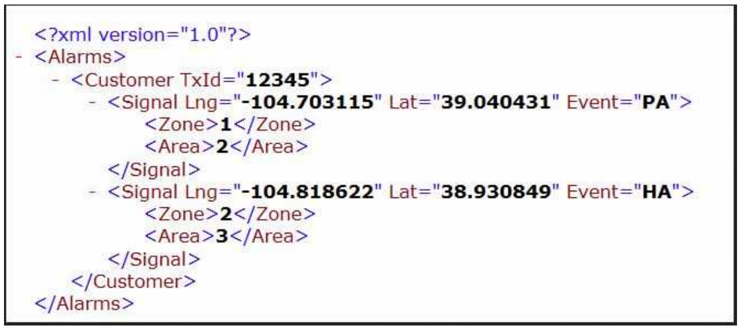
FTP Example

Since the FTP server looks at two directories, XML and binary, the following example provides two example scenarios:

- XML - FTP signal
- Binary - FTP video

FTP Signal

The following is an example of an FTP signal:

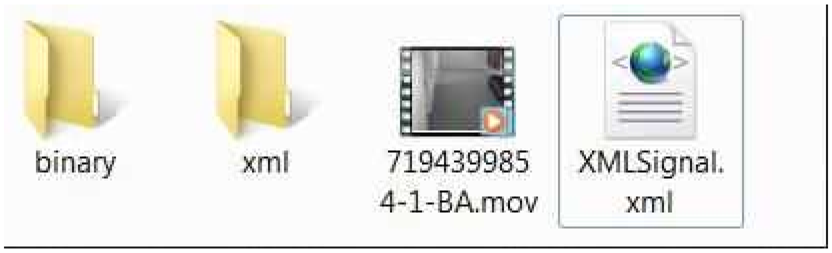


```
<?xml version="1.0"?>
- <Alarms>
  - <Customer TxId="12345">
    - <Signal Lng="-104.703115" Lat="39.040431" Event="PA">
        <Zone>1</Zone>
        <Area>2</Area>
      </Signal>
    - <Signal Lng="-104.818622" Lat="38.930849" Event="HA">
        <Zone>2</Zone>
        <Area>3</Area>
      </Signal>
    </Customer>
</Alarms>
```

XML Signal

FTP Video

Use the following filename to store video information:
• 7194399854-1-BA.mov

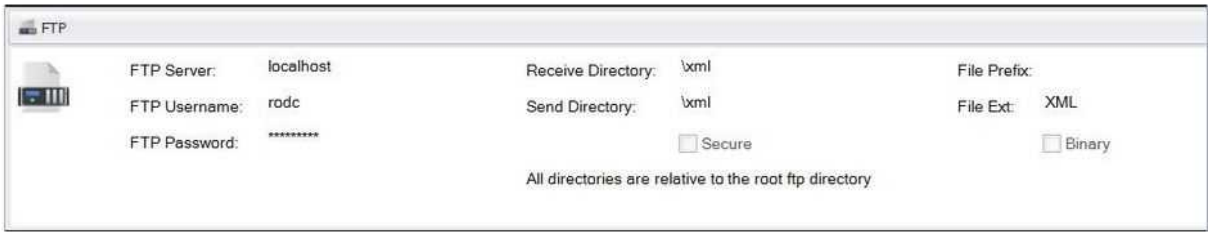


FTP Directories

Connector

Set up both the XML and binary connectors as follows:

XML



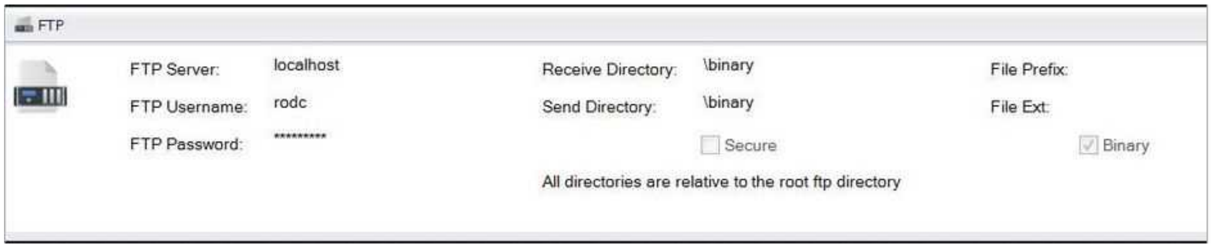
FTP

FTP Server:	localhost	Receive Directory:	\xml	File Prefix:	
FTP Username:	rodc	Send Directory:	\xml	File Ext:	XML
FTP Password:	*********		☐ Secure		☐ Binary

All directories are relative to the root ftp directory

XML Connector Setup example

Binary

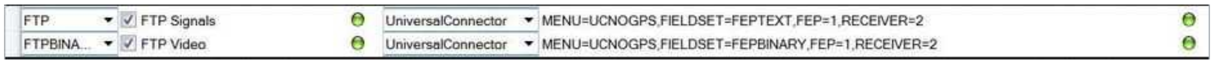


FTP Server: localhost
FTP Username: rodc
FTP Password: *********
Receive Directory: \binary
Send Directory: \binary
☐ Secure
All directories are relative to the root ftp directory
File Prefix:
File Ext:
✓ Binary

Binary Connector Setup example

Line Drivers

Set up the FTP Line Drivers as follows:

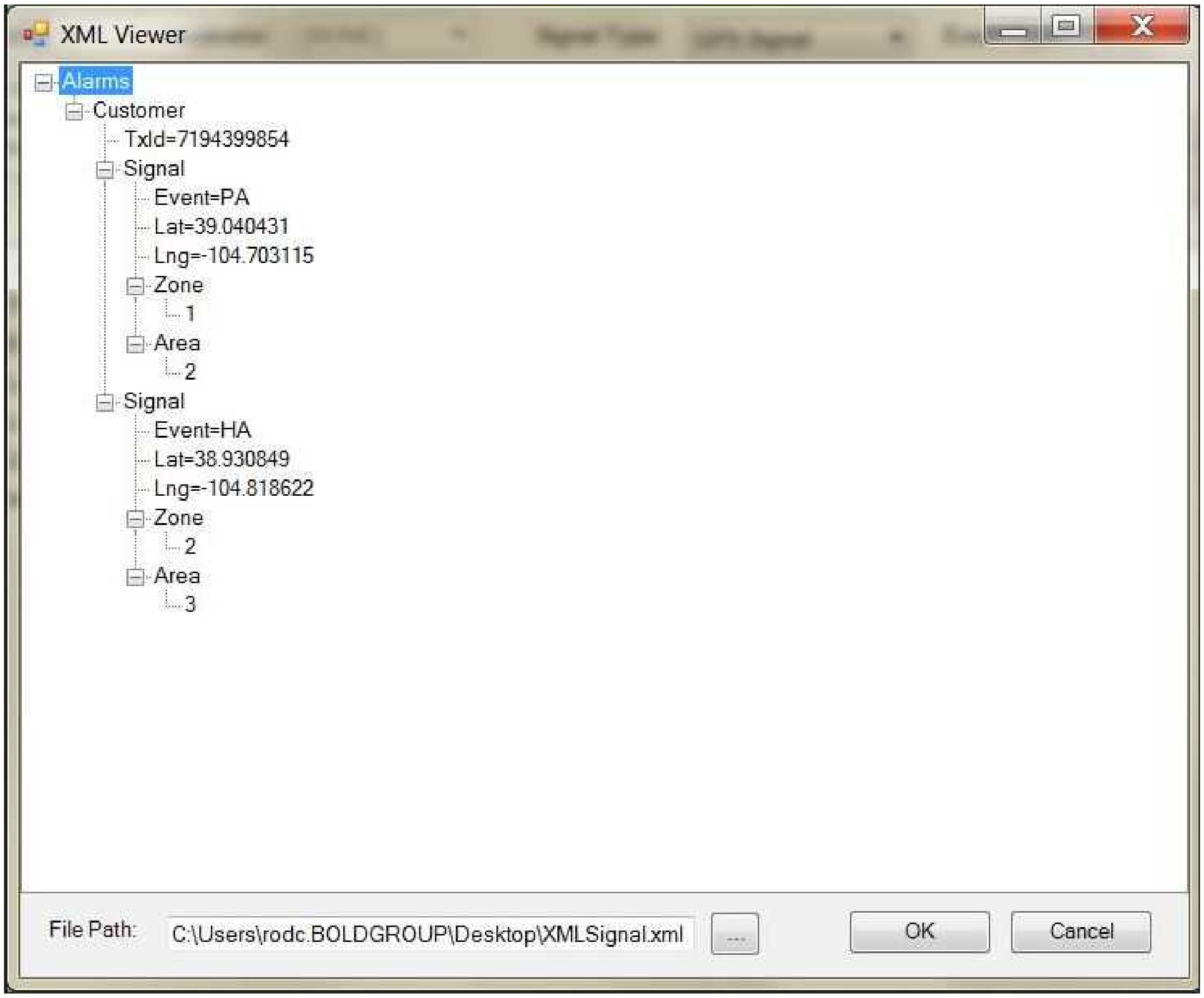


FTP	✓ FTP Signals	●	UniversalConnector	MENU=UCNOGPS,FIELDSET=FEPTEXT,FEP=1,RECEIVER=2	●
FTPBINA...	✓ FTP Video	●	UniversalConnector	MENU=UCNOGPS,FIELDSET=FEPBINARY,FEP=1,RECEIVER=2	●

FTP Line Drivers

Data Mapping

Use XML data mapping for FTP text.

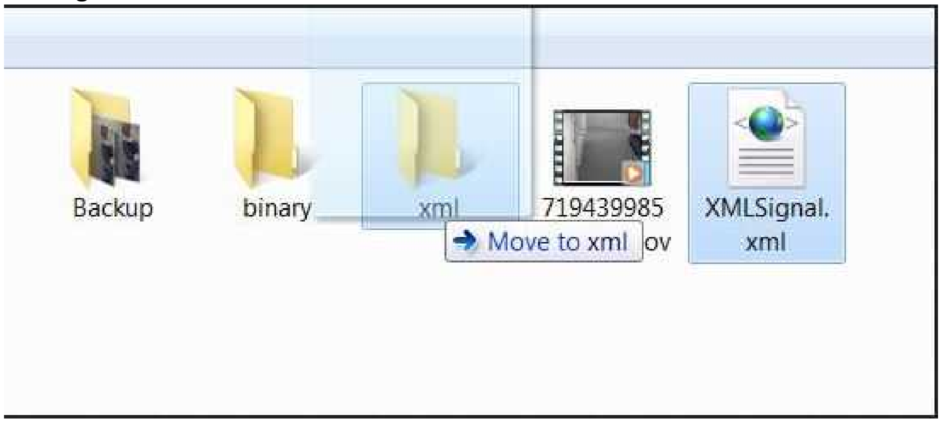

XML Viewer

Alarms
Customer
TxId=7194399854
Signal
Event=PA
Lat=39.040431
Lng=-104.703115
Zone
1
Area
2
Signal
Event=HA
Lat=38.930849
Lng=-104.818622
Zone
2
Area
3

File Path: C:\Users\rodc.BOLDGROUP\Desktop\XMLSignal.xml

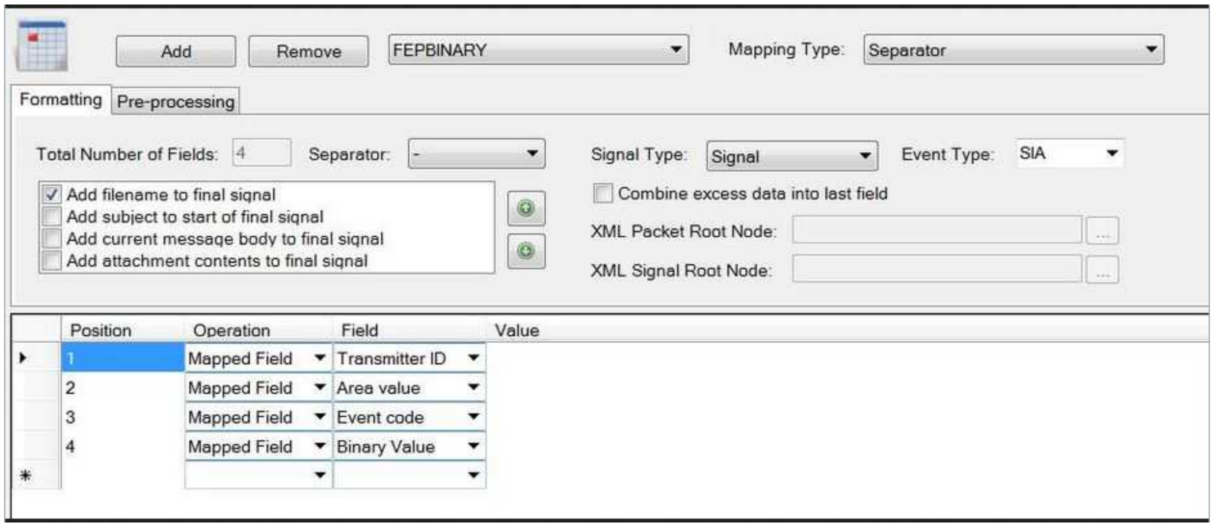
XML Viewer

The FTP format allows you to send multiple signals in one packet. The following example generates two signals in Manitou:



Move XML file

The following is an example of the FTP binary data mapping used for FTP video:


Add | Remove | FEPBINARY | Mapping Type: Separator

Formatting | Pre-processing

Total Number of Fields: 4 Separator: - Signal Type: Signal Event Type: SIA

- [x] Add filename to final signal
- [ ] Add subject to start of final signal
- [ ] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Combine excess data into last field

XML Packet Root Node:

XML Signal Root Node:

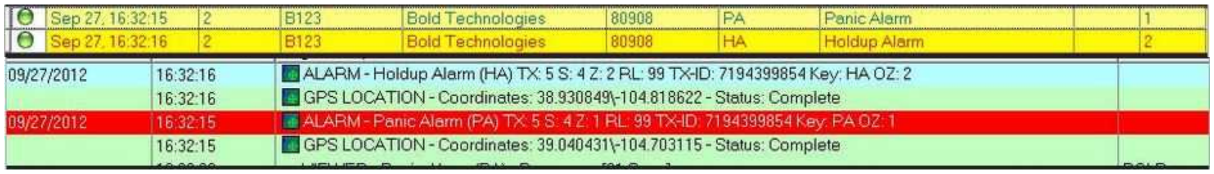
Position	Operation	Field	Value
1	Mapped Field	Transmitter ID	
2	Mapped Field	Area value	
3	Mapped Field	Event code	
4	Mapped Field	Binary Value	
*			

Binary Data Mapping example

Manitou Alarm

XML Text

Moving the file into the XML directory allows the UniversalConnector to download and process the file using the FTP. Manitou then generates the following alarms:

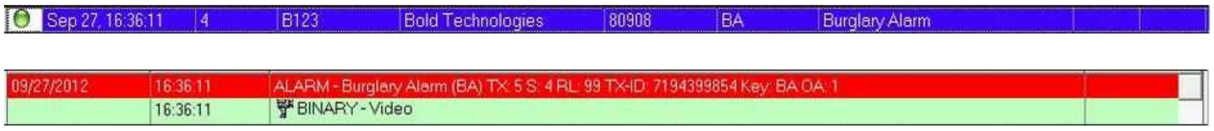

●	Sep 27, 16:32:15	2	B123	Bold Technologies	80908	PA	Panic Alarm	1
●	Sep 27, 16:32:16	2	B123	Bold Technologies	80908	HA	Holdup Alarm	2

09/27/2012	16:32:16	ALARM - Holdup Alarm (HA) TX: 5 S: 4 Z: 2 RL: 99 TX-ID: 7194399854 Key: HA OZ: 2
	16:32:16	GPS LOCATION - Coordinates: 38.930849\-104.818622 - Status: Complete
09/27/2012	16:32:15	ALARM - Panic Alarm (PA) TX: 5 S: 4 Z: 1 RL: 99 TX-ID: 7194399854 Key: PA OZ: 1
	16:32:15	GPS LOCATION - Coordinates: 39.040431\-104.703115 - Status: Complete

FTP Generated Alarm examples

Binary Video

Moving the video file into the binary directory generates a Burglary Alarm with video.


●	Sep 27, 16:36:11	4	B123	Bold Technologies	80908	BA	Burglary Alarm

09/27/2012	16:36:11	ALARM - Burglary Alarm (BA) TX: 5 S: 4 RL: 99 TX-ID: 7194399854 Key: BA OA: 1
	16:36:11	BINARY - Video

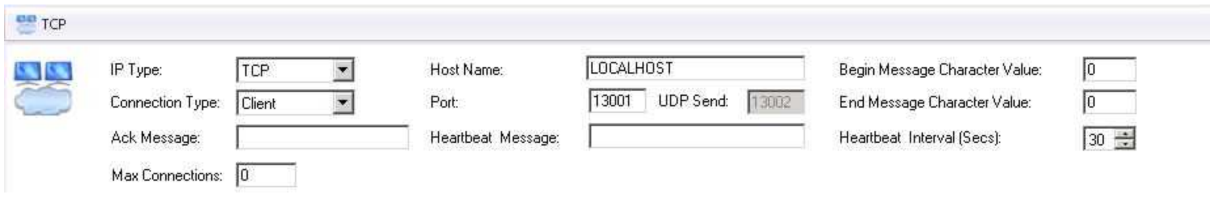
Binary Generated Alarm examples

Alarm Video Clip

TCP

Adding a TCP Connector

The TCP Connector allows the UniversalConnector to accept raw data connections.

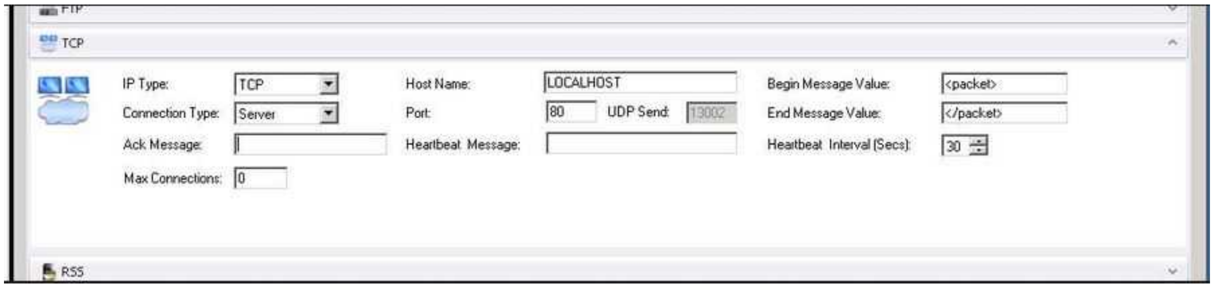


IP Type:	TCP	Host Name:	LOCALHOST	Begin Message Character Value:	0
Connection Type:	Client	Port:	13001 UDP Send: 13002	End Message Character Value:	0
Ack Message:		Heartbeat Message:		Heartbeat Interval (Secs):	30
Max Connections:	0				

TCP Configuration

Using XML Tokens

You may also add an XML token for the beginning and ending values on a TCP Connector.

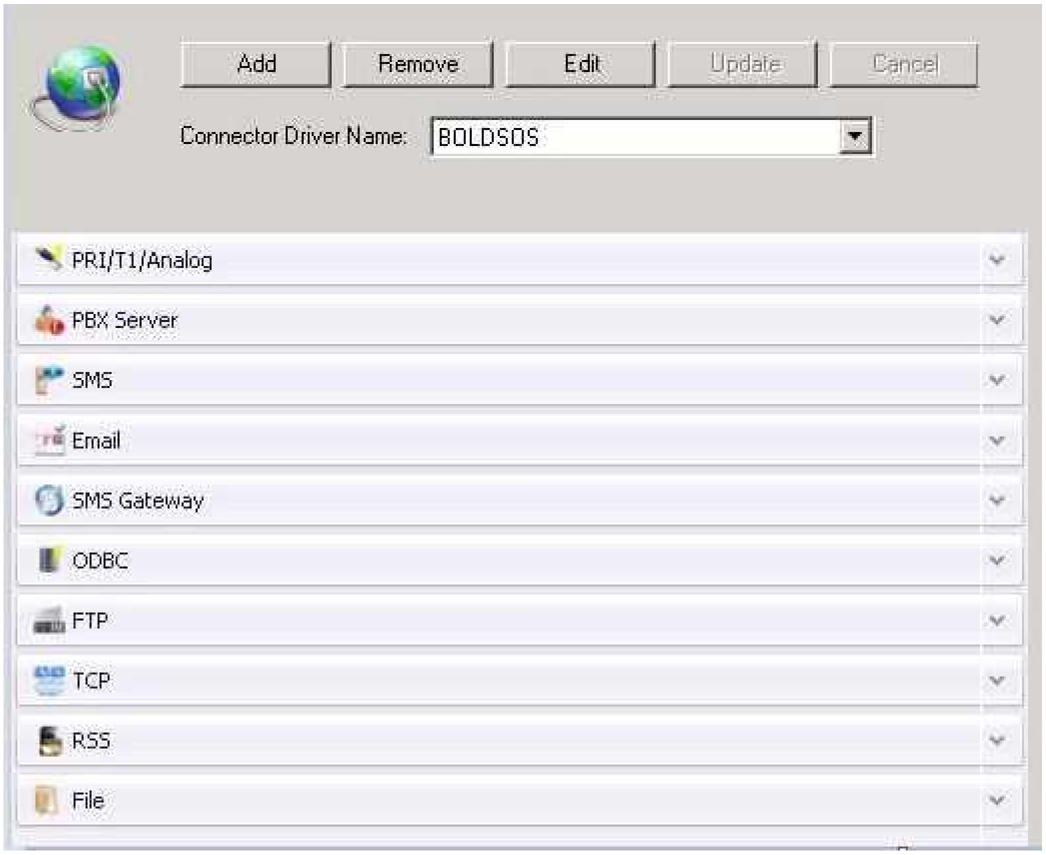


IP Type:	TCP	Host Name:	LOCALHOST	Begin Message Value:	<packet>
Connection Type:	Server	Port:	80 UDP Send: 13002	End Message Value:	</packet>
Ack Message:		Heartbeat Message:		Heartbeat Interval (Secs):	30
Max Connections:	0				

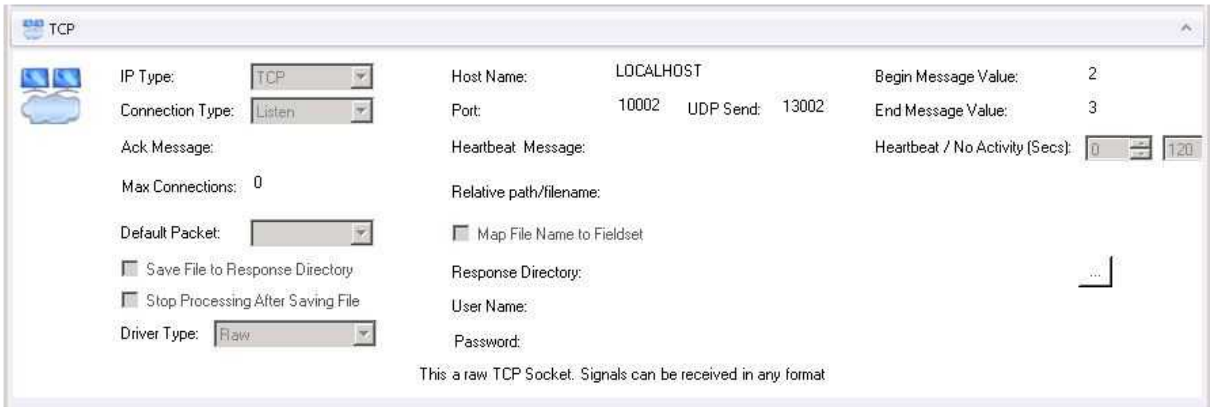
TCP Config., XML Tokens

Perform the following steps to add a TCP Connector:

1. Click "Connector" under the UniversalConnector section of the Node Tree.
Result: the following window displays:

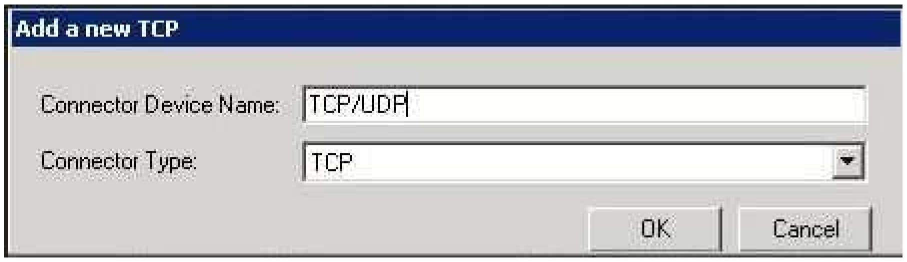


2. Click the down arrow to expand the TCP area of the window as shown in the following screenshot:


TCP
Save File to Response Directory
Stop Processing After Saving File
Map File Name to Fieldset

IP Type: TCP
Connection Type: Listen
Ack Message:
Max Connections: 0
Default Packet:
Driver Type: Raw
Host Name: LOCALHOST
Port: 10002 UDP Send: 13002
Heartbeat Message:
Relative path/filename:
Response Directory:
User Name:
Password:
Begin Message Value: 2
End Message Value: 3
Heartbeat / No Activity (Secs): 0 120
This a raw TCP Socket. Signals can be received in any format

3. Click "**Add**".
Result: the "Add a new TCP" dialog displays as shown in the following screenshot:



Connector Device Name: TCP/UDP

Connector Type: TCP

4. Enter a device name into the "Connector Device Name:" field.
5. Confirm that the "Connector Type:" field displays as "TCP". If it does not, select "TCP" from the dropdown list.
6. Click "**OK**".

Result: the "Add a new TCP" window closes and the fields on the TCP area of the window now display as populated with the default settings.

Note: you may need to modify the default settings depending on the hardware components you have installed. Use the dropdown arrows to access additional selections available for each field.

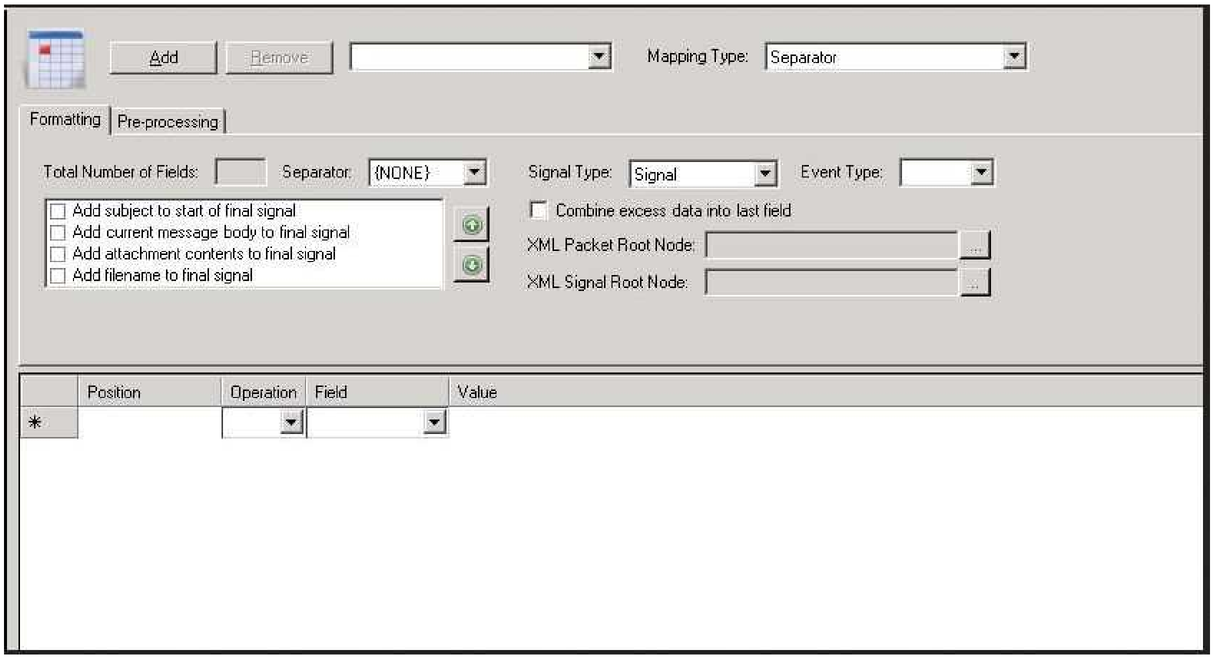
Data Mapping

In order for the communication to be properly identified and submitted to Manitou, you must define the correct data mapping in the Data Mapping form.

Perform the following steps to define the correct data mapping:

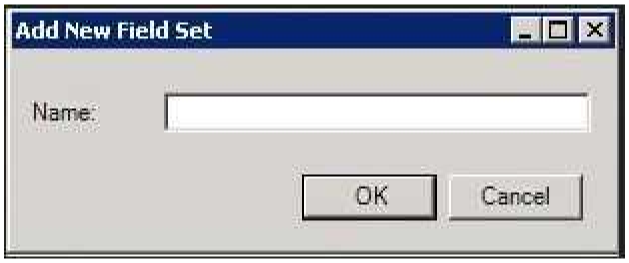
1. Click "Data Mapping" under UniversalConnector on the Node Tree.

Result: the Data Mapping form displays as shown in the following screenshot:



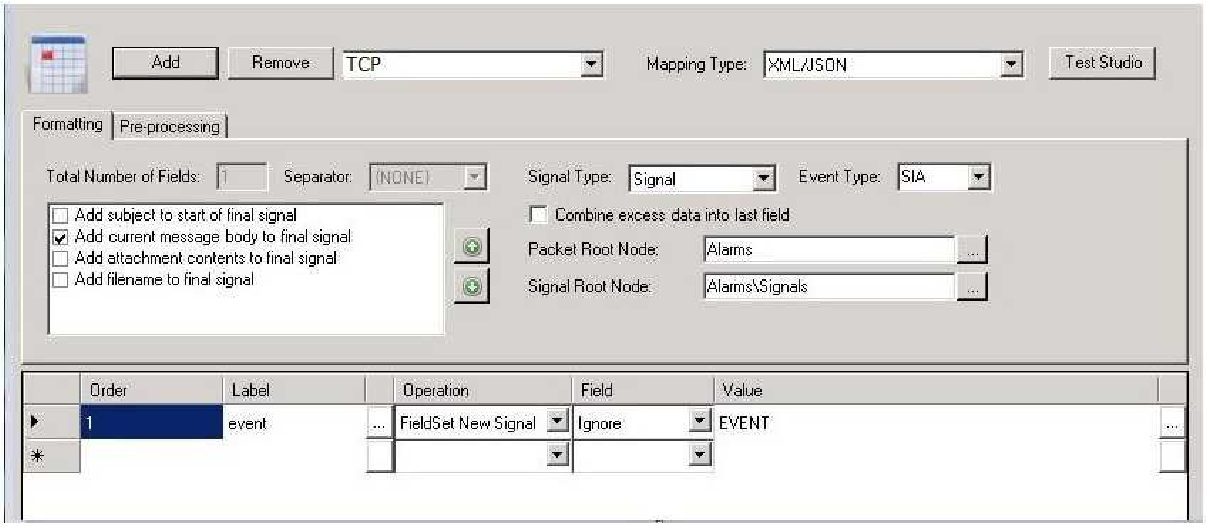
2. Select "TCP" from the dropdown menu directly to the right of the Remove button.
3. Click "**Add**".

Result: the "Add New Field Set" dialog displays as shown in the following screenshot:



4. Enter a name into the "Name:" field, and then click "**OK**".

Result: the "Add New Field Set" window closes and the system returns you to the Data Mapping form as shown in the following screenshot:


Add | Remove | TCP | Mapping Type: XML/JSON | Test Studio

Formatting | Pre-processing

Total Number of Fields: 1 | Separator: (NONE) | Signal Type: Signal | Event Type: SIA

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [ ] Combine excess data into last field

Packet Root Node: Alarms

Signal Root Node: Alarms\Signals

Order	Label	Operation	Field	Value
1	event	FieldSet New Signal	Ignore	EVENT

5. Select an option from the "Mapping Type:" dropdown menu.

Note: your options are: Separator, Position, Label/Separator, and XML/JSON.

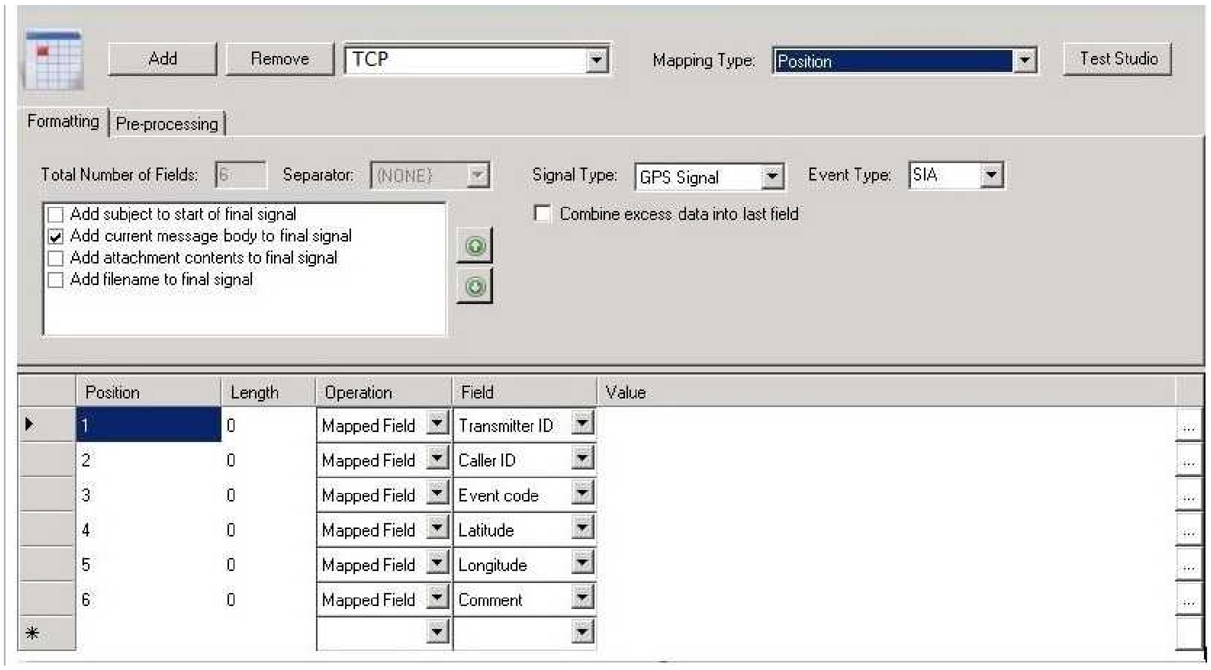
6. Select an option from the "Separator:" dropdown menu.
7. Select an option from the "Signal Type:" dropdown menu.

Note: your options are: Signal, GPS Signal, and Telemetry. For more information on these signal types, please refer to the document entitled "Field Descriptions".

8. Select an option from the "Event Type:" dropdown menu.

Note: your options are: SYS, SIA, and CID.

9. If you want to add part of the message to the final signal, select any of the four checkboxes.
10. If you want, select the "Combine excess data into the last field" checkbox.
11. In the Data Parameters section in the bottom portion of the screen, designate the order you want the information to be parsed into Manitou in the "Position" column.



TCP

Mapping Type: Position

Total Number of Fields: 6 Separator: (NONE) Signal Type: GPS Signal Event Type: SIA

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [ ] Combine excess data into last field

Position	Length	Operation	Field	Value
1	0	Mapped Field	Transmitter ID	
2	0	Mapped Field	Caller ID	
3	0	Mapped Field	Event code	
4	0	Mapped Field	Latitude	
5	0	Mapped Field	Longitude	
6	0	Mapped Field	Comment	

Note: the column labels in the Data Parameters area change depending on the Mapping Type you selected.

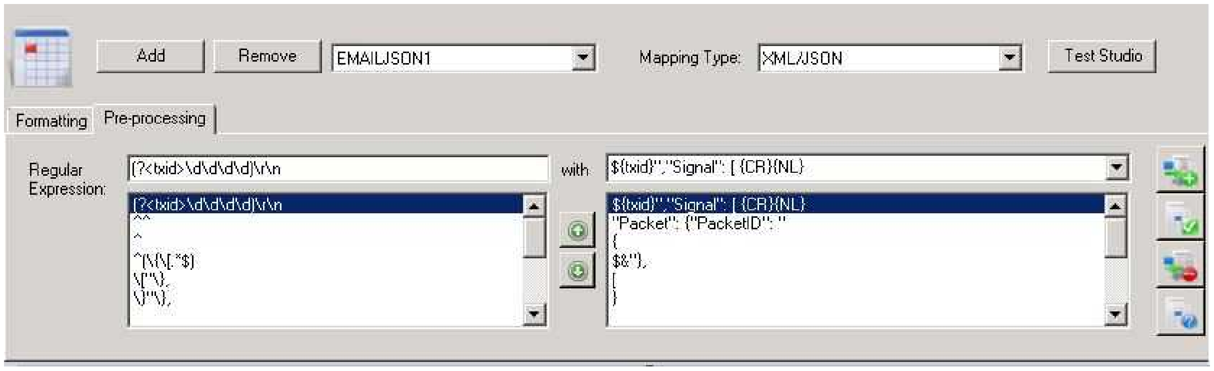
Data Mapping, Pre-processing

Defining Expressions in the Pre-processing Tab

The Pre-processing tab enables you to anticipate and prevent some common processing issues before they occur. For example, if a comma was designated as the separator on the Formatting tab, but there are instances where a semi-colon might come through the received signal as the separator, you can avoid this issue before it occurs.

Perform the following steps to use the Pre-processing tab to avoid common processing issues:

1. Click the Pre-processing tab on the Data Mapping form.
Result: the Pre-processing tab displays as shown in the following screenshot:



EMAILJSON1

Mapping Type: XML/JSON

Regular Expression: (?<txid>\d\d\d\d)\r\n

with ${txid}","Signal": [{CR}{NL}

```
(?<txid>\d\d\d\d)\r\n
^^
^
^(\{\[.*$)
\["\},
\}"\},
```

```
${txid}","Signal": [ {CR}{NL}
"Packet": {"PacketID": "
{
$&"},
[
}
```

2. In the "Regular Expression:" field, enter the symbol or letters used as a common expression in the signals you receive.
3. Enter the interpretation option you want for the common expression in the right-hand field.

The buttons on the right-hand side of the window perform the following functions:

Add regular expression

Update regular expression

Remove regular expression

Test Studio

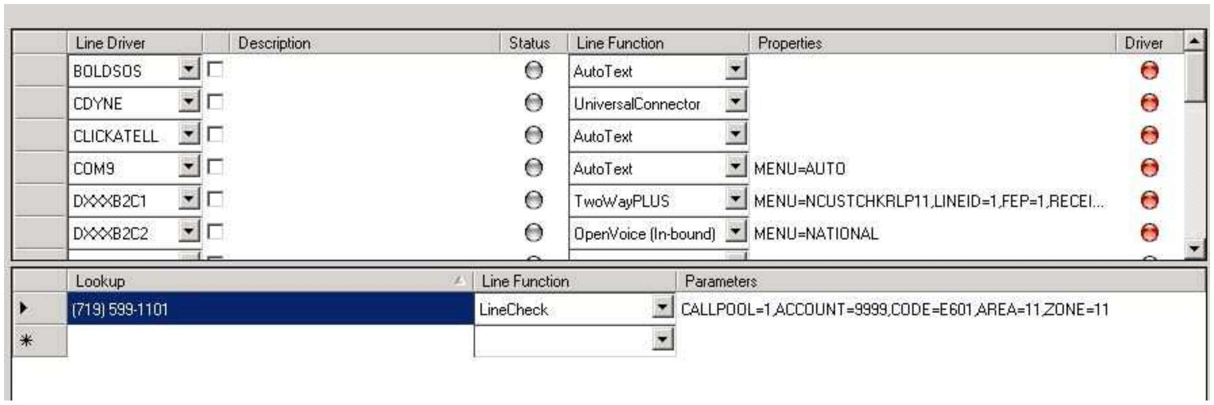
Line Drivers

Once you have created a TCP Connector and mapped it, you must format the Line Drivers.

Note: the FEP and associated Receivers must be correctly installed and configured prior to formatting Line Drivers. To check the installation of the FEPs and Receivers, please refer to the appropriate Supervisor Workstation manual for your version of Manitou.

Perform the following steps to format the Line Drivers:

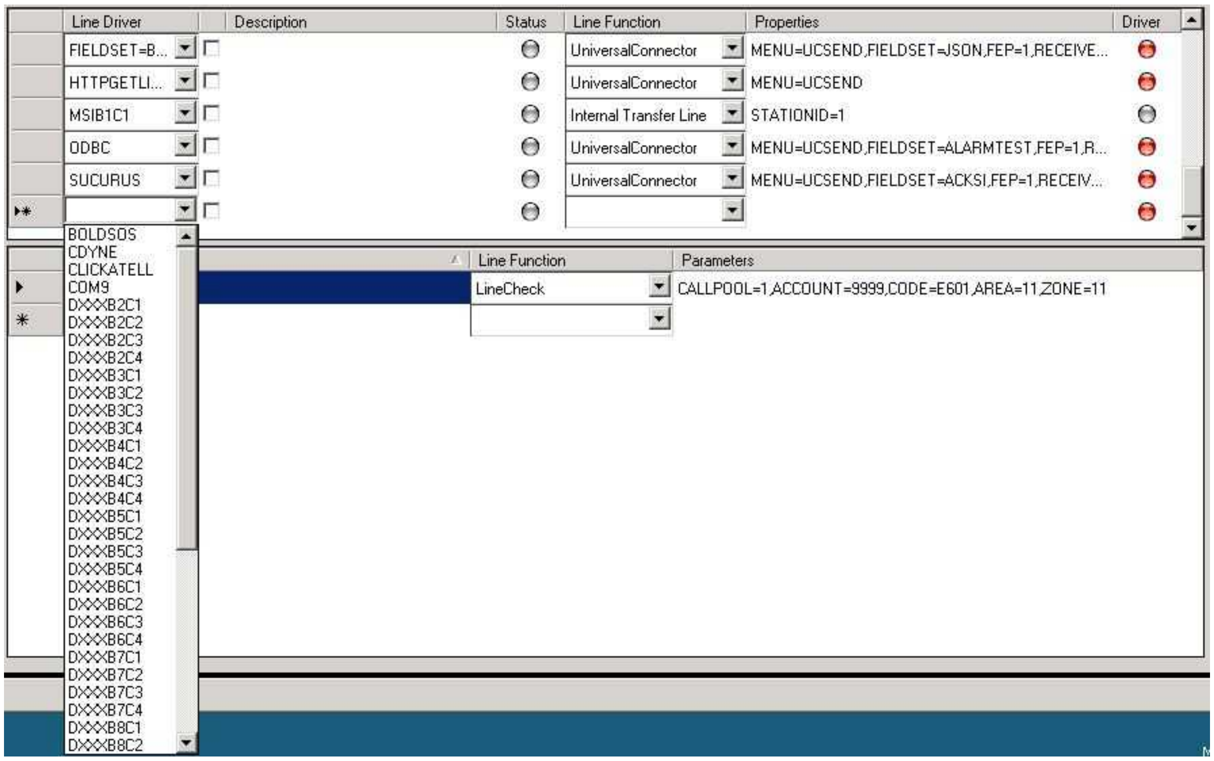
1. Click "Line Drivers" under Administration in the Node Tree.
Result: the Line Drivers form displays shown in the following screenshot:



Line Driver		Description	Status	Line Function	Properties	Driver
BOLDSOS	☐		●	AutoText		🔴
CDYNE	☐		●	UniversalConnector		🔴
CLICKATELL	☐		●	AutoText		🔴
COM9	☐		●	AutoText	MENU=AUTO	🔴
DXXXB2C1	☐		●	TwoWayPLUS	MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	🔴
DXXXB2C2	☐		●	OpenVoice (In-bound)	MENU=NATIONAL	🔴

Lookup	Line Function	Parameters
(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

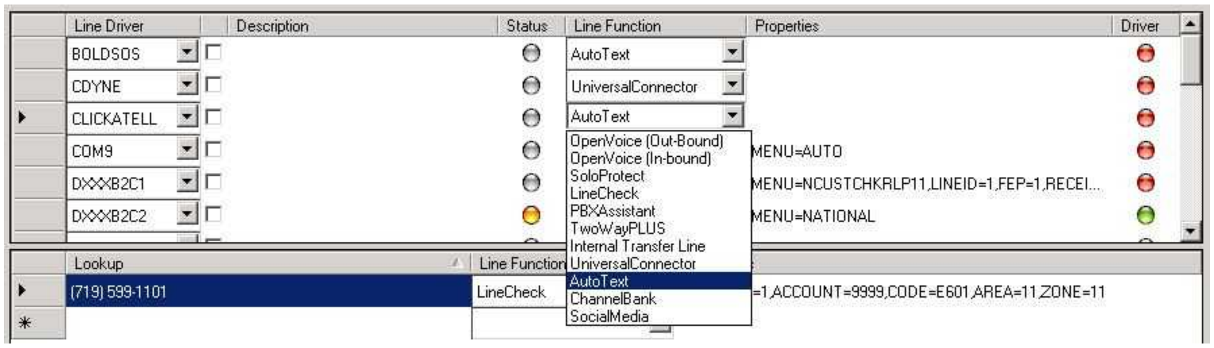
2. In the "Line Driver" column, select the Line Driver from the pulldown list in the next open row (indicated with an asterisk) as shown in the following screenshot:



Line Driver		Description	Status	Line Function	Properties	Driver
FIELDSET=B...	☐		●	UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	🔴
HTTPGETLI...	☐		●	UniversalConnector	MENU=UCSEND	🔴
MSIB1C1	☐		●	Internal Transfer Line	STATIONID=1	●
ODBC	☐		●	UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	🔴
SUCURUS	☐		●	UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	🔴
	☐		●			🔴

Line Function	Parameters
LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

3. If you want the Line Driver to be designated as active, select the checkbox to the right of it. If you do not want it to be active, leave the checkbox deselected.
4. Enter a description into the "Description" field.
5. Select the type of "Line Function" you want from the pulldown menu as shown in the following screenshot:

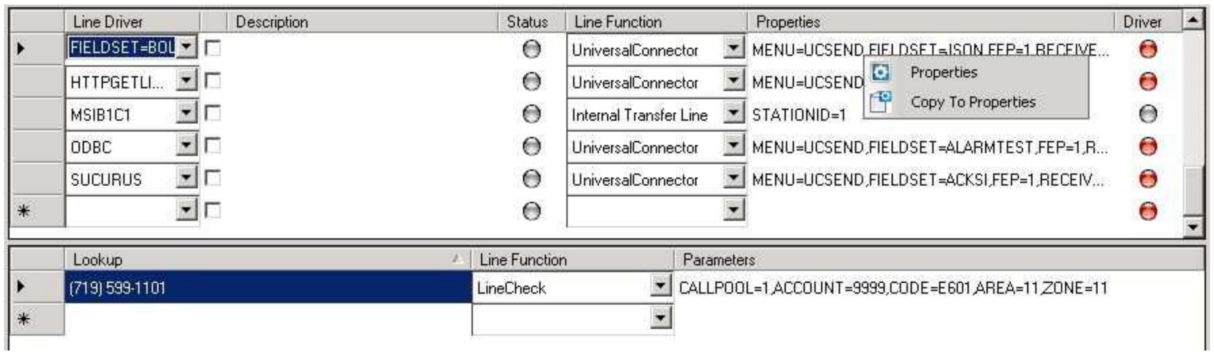


Line Driver	Description	Status	Line Function	Properties	Driver
BOLDSOS	☐	●	AutoText		●
CDYNE	☐	●	UniversalConnector		●
CLICKATELL	☐	●	AutoText		●
COM9	☐	●		MENU=AUTO	●
DXXXB2C1	☐	●		MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	●
DXXXB2C2	☐	●		MENU=NATIONAL	●

Lookup	Line Function	
(719) 599-1101	LineCheck	=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

Establish Properties

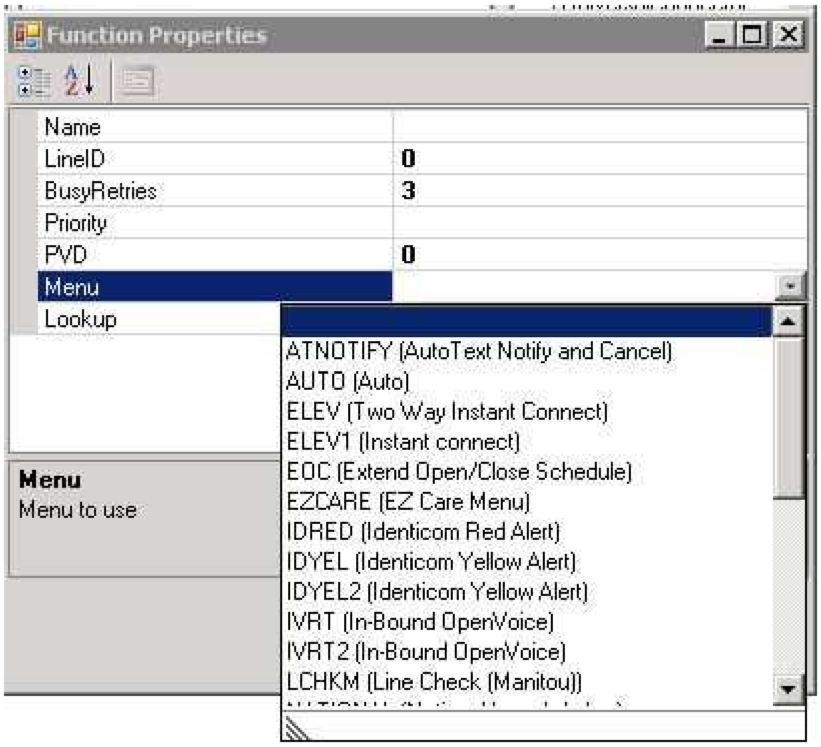
1. To enter property information for a Line Driver, select the Driver you want. Then, right-click in the "Properties" column and select Properties from the menu that displays. You also have the option of copying properties from an already existing Line Driver by selecting the "Copy to Properties" option as shown in the following screenshot:



Line Driver	Description	Status	Line Function	Properties	Driver
FIELDSET=BOL	☐	●	UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	●
HTTPGETLI...	☐	●	UniversalConnector	MENU=UCSEND	●
MSIB1C1	☐	●	Internal Transfer Line	STATIONID=1	●
ODBC	☐	●	UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	●
SUCURUS	☐	●	UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	●

Lookup	Line Function	Parameters
(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

Result: the Function Properties window displays as shown in the following screenshot:



Name	
LineID	0
BusyRetries	3
Priority	
PVD	0
Menu	
Lookup	

Menu
Menu to use

ATNOTIFY (AutoText Notify and Cancel)
AUTO (Auto)
ELEV (Two Way Instant Connect)
ELEV1 (Instant connect)
EOC (Extend Open/Close Schedule)
EZCARE (EZ Care Menu)
IDRED (Identicom Red Alert)
IDYEL (Identicom Yellow Alert)
IDYEL2 (Identicom Yellow Alert)
IVRT (In-Bound OpenVoice)
IVRT2 (In-Bound OpenVoice)
LCHKM (Line Check (Manitou))

Note: the function list items that display on the Function Properties menu depend on the Line Driver you selected.

2. Enter the values you want in the right column to correspond to the function list on the left. Some functions have dropdown menus (as shown in the screenshot above). To access a dropdown menu, click at the edge of the right column until a down arrow displays. Click the arrow, and scroll through the list to select the item you want.
3. Click "**OK**" when finished, and then click "**Save**".

RSS

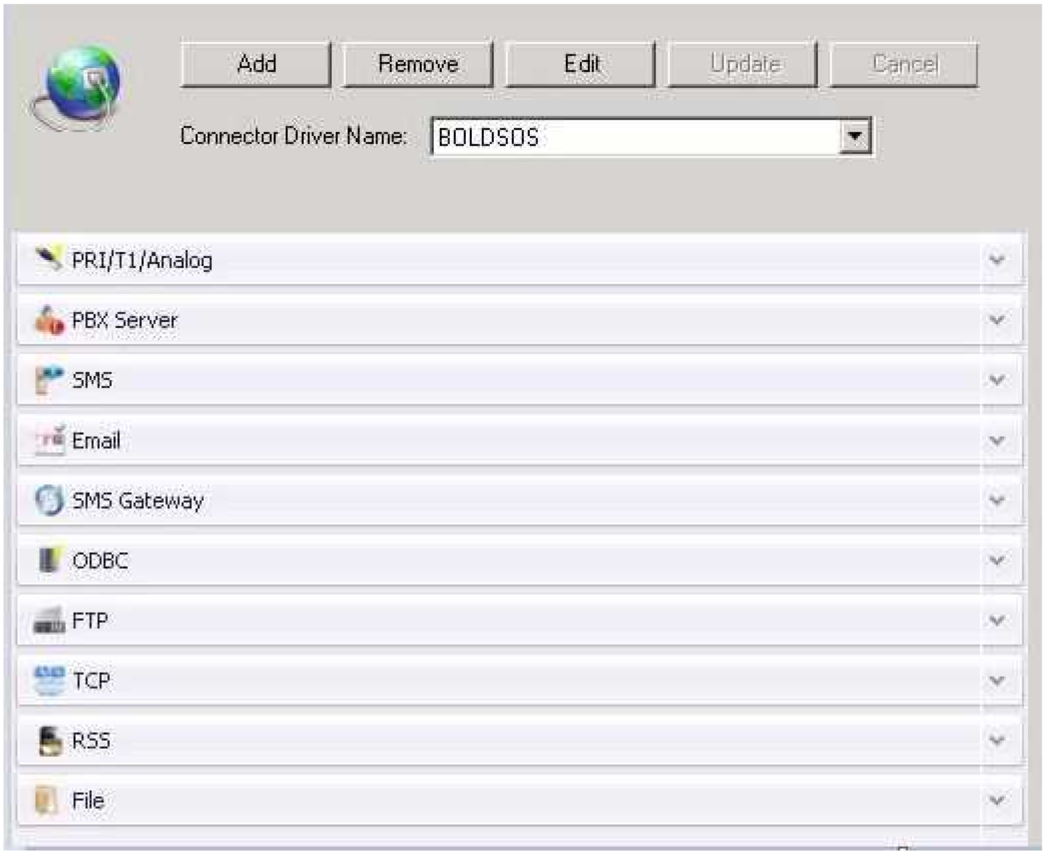
Adding an RSS Connector

The RSS Connector enables the UniversalConnector to monitor, pick up, parse, and send RSS feeds into Manitou as alarms.

Perform the following steps to add an RSS Connector:

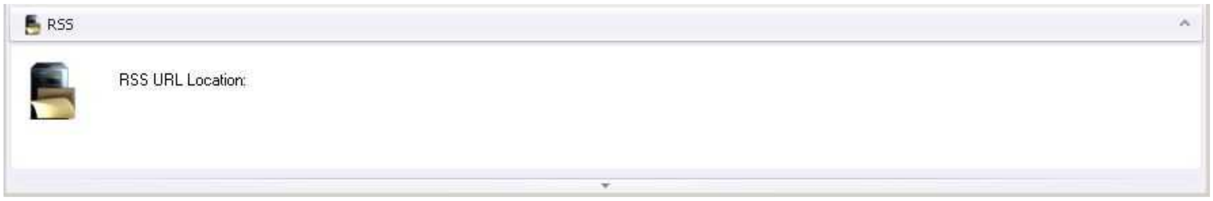
1. Click "Connector" under the UniversalConnector section of the Node Tree.

Result: the following window displays:



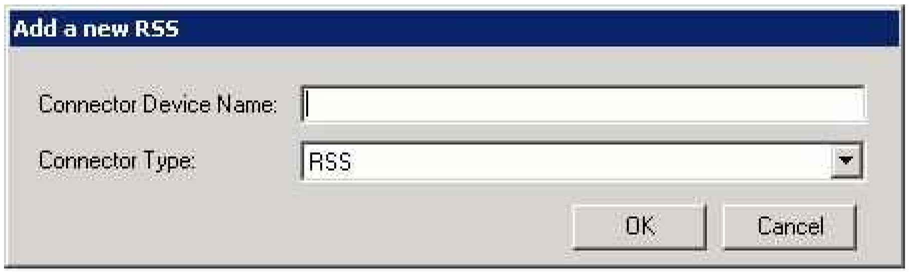
Connector Driver Name: BOLDSOS

2. Click the down arrow to expand the RSS area of the window as shown in the following screenshot:



RSS URL Location:

3. Click "**Add**".
Result: the "Add a new RSS" dialog displays as shown in the following screenshot:



Add a new RSS

Connector Device Name:

Connector Type: RSS

4. Enter a device name into the "Connector Device Name:" field.
5. Confirm that the "Connector Type:" field displays as "RSS". If it does not, select "RSS" from the dropdown list.
6. Click "**OK**".

Result: the "Add a new RSS" window closes and the fields on the RSS area of the window now display as populated with the default settings.
Note: you may need to modify the default settings depending on the hardware components you have installed. Use the dropdown arrows to access additional selections available for each field.

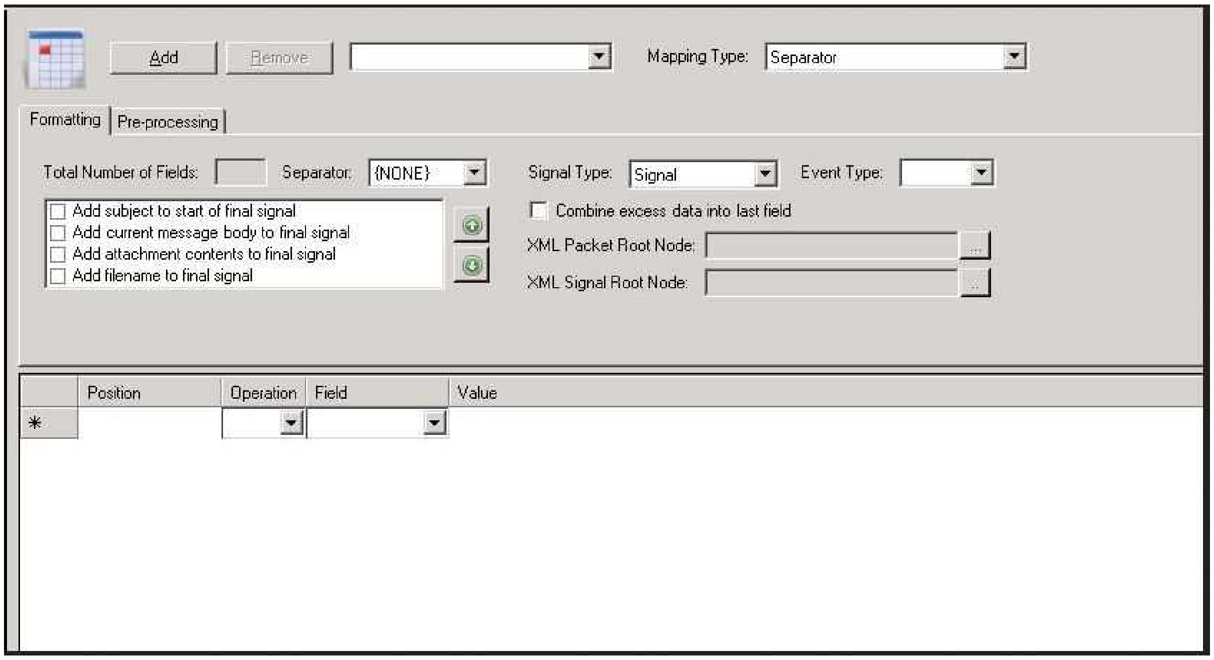
Data Mapping

In order for the communication to be properly identified and submitted to Manitou, you must define the correct data mapping in the Data Mapping form.

Perform the following steps to define the correct data mapping:

1. Click "Data Mapping" under UniversalConnector on the Node Tree.

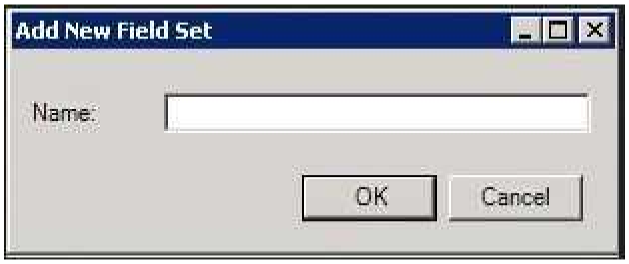
Result: the Data Mapping form displays as shown in the following screenshot:


Add
Remove
Formatting
Pre-processing
Add subject to start of final signal
Add current message body to final signal
Add attachment contents to final signal
Add filename to final signal
Combine excess data into last field
Total Number of Fields:
XML Packet Root Node:
XML Signal Root Node:
Event Type:
Position
Operation
Field
Value

Mapping Type: Separator
Separator: {NONE}
Signal Type: Signal

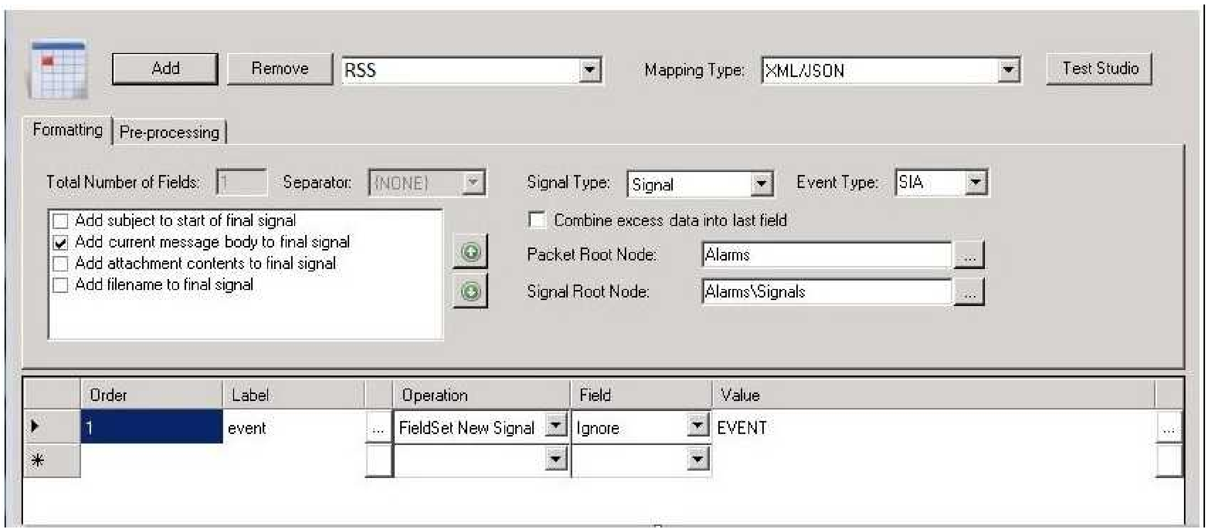
2. Select "RSS" from the dropdown menu directly to the right of the Remove button.
3. Click "**Add**".

Result: the "Add New Field Set" dialog displays as shown in the following screenshot:


Add New Field Set

Name:

4. Enter a name into the "Name:" field, and then click "**OK**".
Result: the "Add New Field Set" window closes and the system returns you to the Data Mapping form as shown in the following screenshot:



RSS

Mapping Type: XML/JSON

Total Number of Fields: 1 Separator: (NONE) Signal Type: Signal Event Type: SIA

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

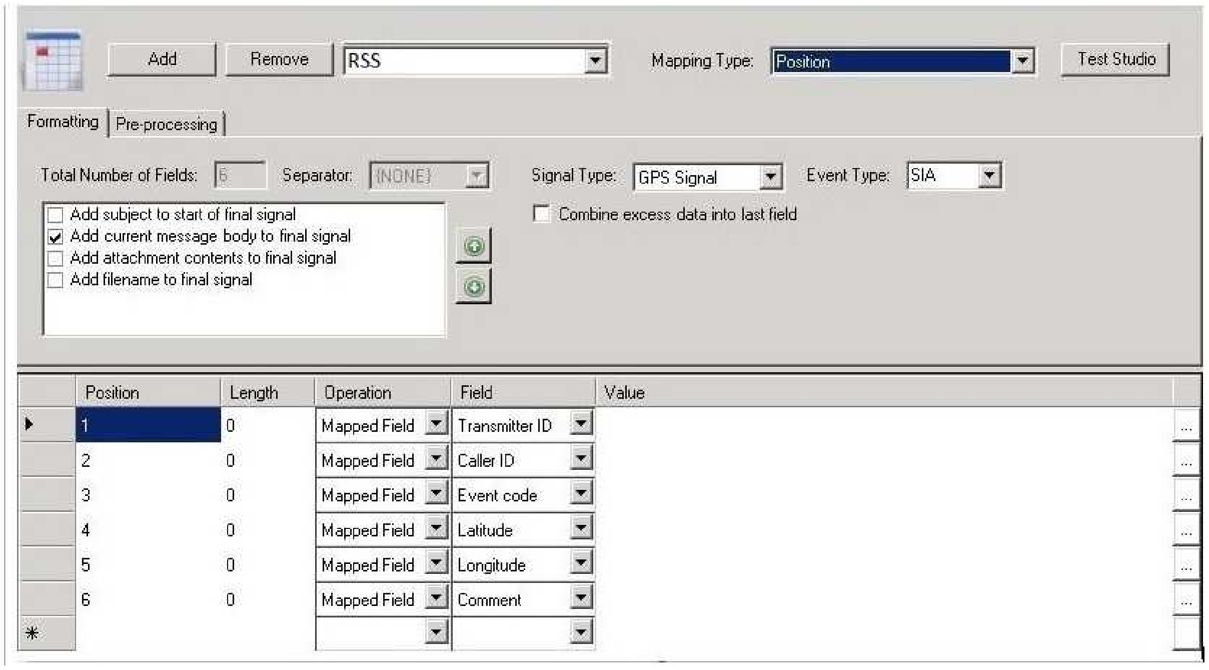
- [ ] Combine excess data into last field

Packet Root Node: Alarms

Signal Root Node: Alarms\Signals

Order	Label	Operation	Field	Value
1	event	FieldSet New Signal	Ignore	EVENT

5. Select an option from the "Mapping Type:" dropdown menu.
Note: your options are: Separator, Position, Label/Separator, and XML/JSON.
6. Select an option from the "Separator:" dropdown menu.
7. Select an option from the "Signal Type:" dropdown menu.
Note: your options are: Signal, GPS Signal, and Telemetry. For more information on these signal types, please refer to the document entitled "Field Descriptions".
8. Select an option from the "Event Type:" dropdown menu.
Note: your options are: SYS, SIA, and CID.
9. If you want to add part of the message to the final signal, select any of the four checkboxes.
10. If you want, select the "Combine excess data into the last field" checkbox.
11. In the Data Parameters section in the bottom portion of the screen, designate the order you want the information to be parsed into Manitou in the "Position" column.



RSS

Mapping Type: Position

Total Number of Fields: 6 Separator: {NONE} Signal Type: GPS Signal Event Type: SIA

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [ ] Combine excess data into last field

Position	Length	Operation	Field	Value
1	0	Mapped Field	Transmitter ID	
2	0	Mapped Field	Caller ID	
3	0	Mapped Field	Event code	
4	0	Mapped Field	Latitude	
5	0	Mapped Field	Longitude	
6	0	Mapped Field	Comment	

Note: the column labels in the Data Parameters area change depending on the Mapping Type you selected.

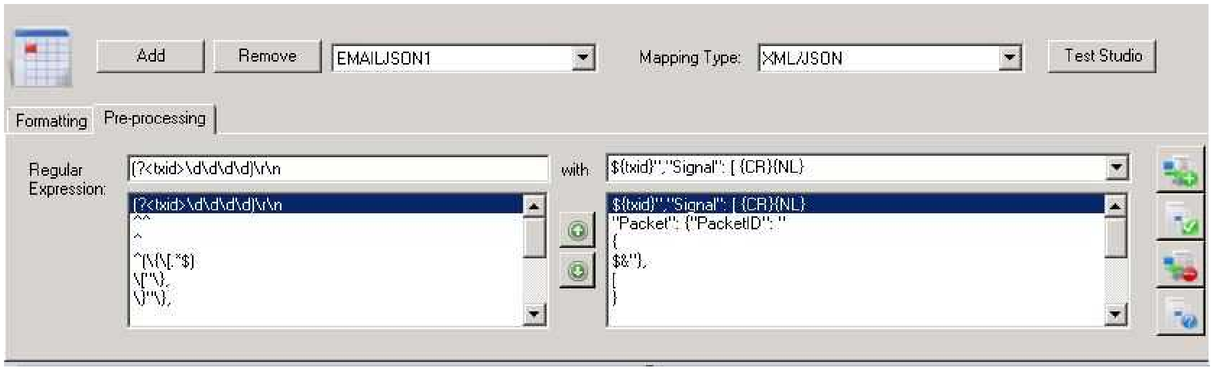
Data Mapping, Pre-processing

Defining Expressions in the Pre-processing Tab

The Pre-processing tab enables you to anticipate and prevent some common processing issues before they occur. For example, if a comma was designated as the separator on the Formatting tab, but there are instances where a semi-colon might come through the received signal as the separator, you can avoid this issue before it occurs.

Perform the following steps to use the Pre-processing tab to avoid common processing issues:

1. Click the Pre-processing tab on the Data Mapping form.
Result: the Pre-processing tab displays as shown in the following screenshot:



EMAILJSON1 Mapping Type: XML/JSON

Regular Expression: (?<txid>\d\d\d\d)\r\n with ${txid}","Signal": [{CR}{NL}

```
(?<txid>\d\d\d\d)\r\n        ${txid}","Signal": [ {CR}{NL}
^^                           "Packet": {"PacketID": "
^                            {
^(\{\[.*$)                   $&"},
\["\},                       [
\}"\},                       }
```

2. In the "Regular Expression:" field, enter the symbol or letters used as a common expression in the signals you receive.
3. Enter the interpretation option you want for the common expression in the right-hand field.

The buttons on the right-hand side of the window perform the following functions:

Add regular expression

Update regular expression

Remove regular expression

Test Studio

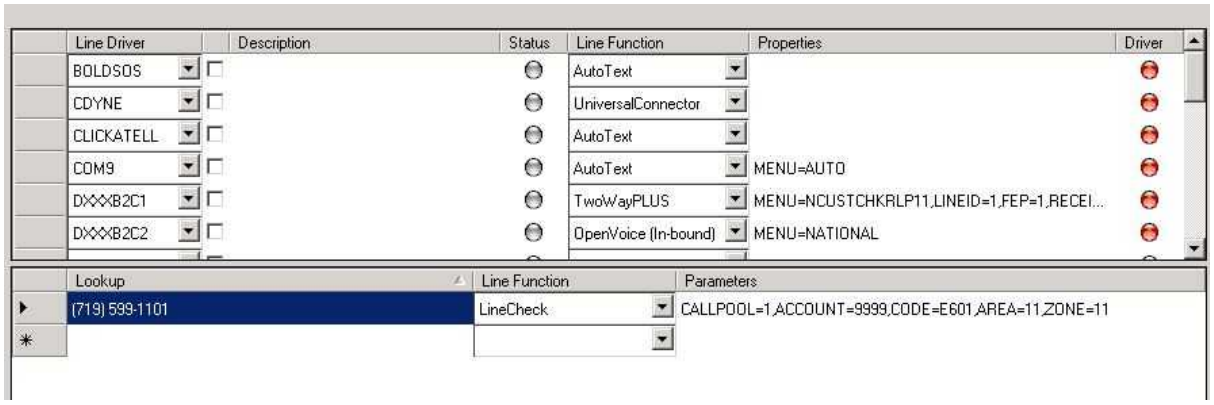
Line Drivers

Once you have created an RSS Connector and mapped it, you must format the Line Drivers.

Note: the FEP and associated Receivers must be correctly installed and configured prior to formatting Line Drivers. To check the installation of the FEPs and Receivers, please refer to the appropriate Supervisor Workstation manual for your version of Manitou.

Perform the following steps to format the Line Drivers:

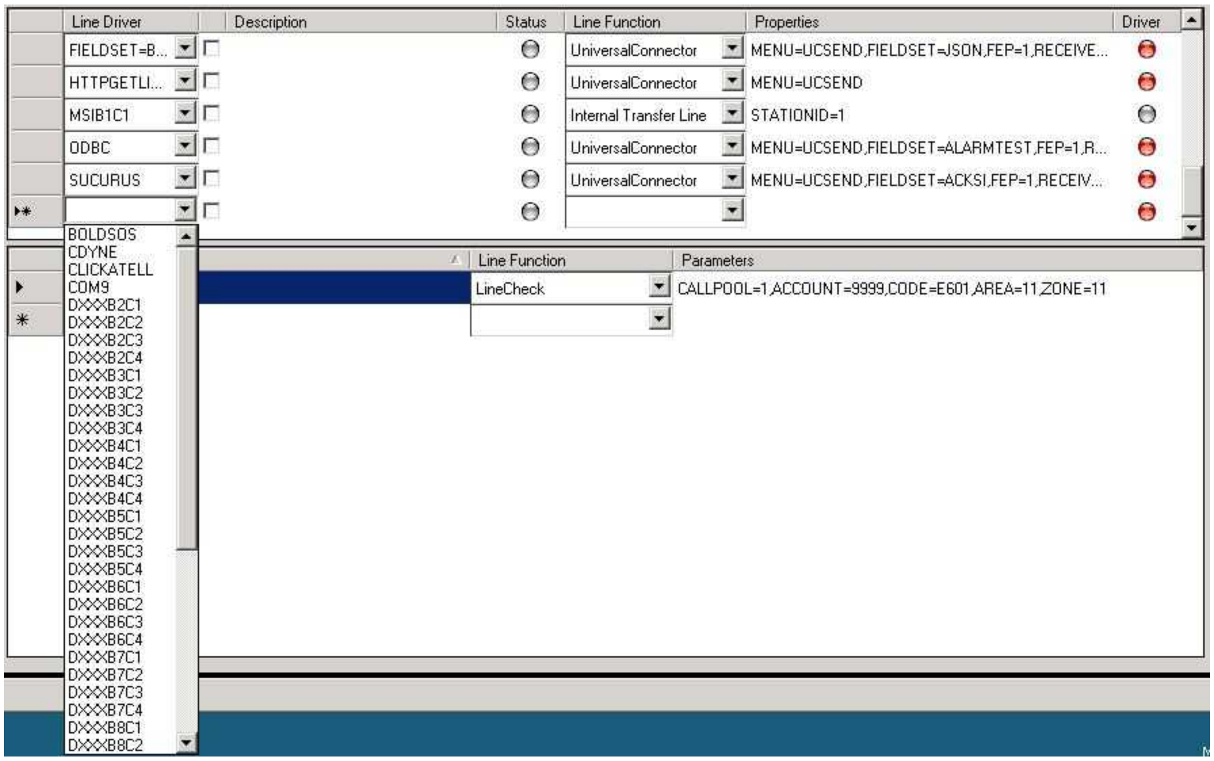
1. Click "Line Drivers" under Administration in the Node Tree.
Result: the Line Drivers form displays shown in the following screenshot:



Line Driver		Description	Status	Line Function	Properties	Driver
BOLDSOS	☐			AutoText		
CDYNE	☐			UniversalConnector		
CLICKATELL	☐			AutoText		
COM9	☐			AutoText	MENU=AUTO	
DXXXB2C1	☐			TwoWayPLUS	MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	
DXXXB2C2	☐			OpenVoice (In-bound)	MENU=NATIONAL	

Lookup	Line Function	Parameters
(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

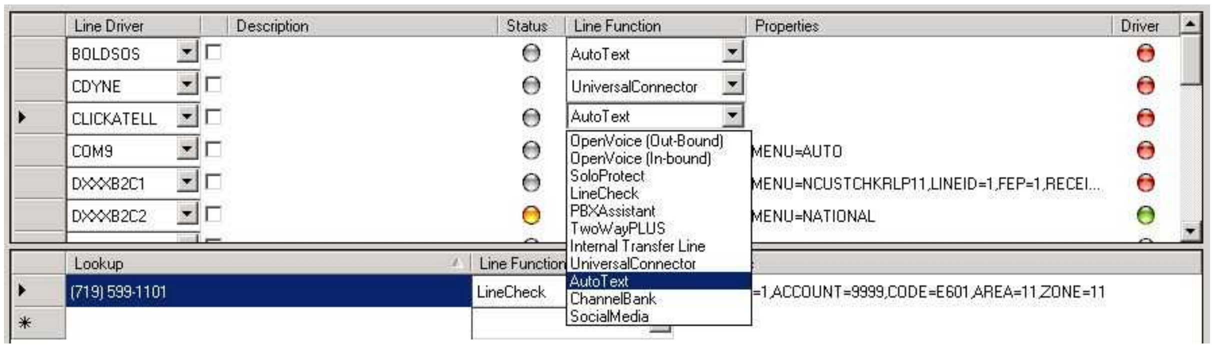
2. In the "Line Driver" column, select the Line Driver from the pulldown list in the next open row (indicated with an asterisk) as shown in the following screenshot:



Line Driver		Description	Status	Line Function	Properties	Driver
FIELDSET=B...	☐			UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	
HTTPGETLI...	☐			UniversalConnector	MENU=UCSEND	
MSIB1C1	☐			Internal Transfer Line	STATIONID=1	
ODBC	☐			UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	
SUCURUS	☐			UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	
	☐					

Line Function	Parameters
LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

3. If you want the Line Driver to be designated as active, select the checkbox to the right of the driver. If you do not want it to be active, leave the checkbox deselected.
4. Enter a description into the "Description" field.
5. Select the type of "Line Function" you want from the pulldown menu as shown in the following screenshot:

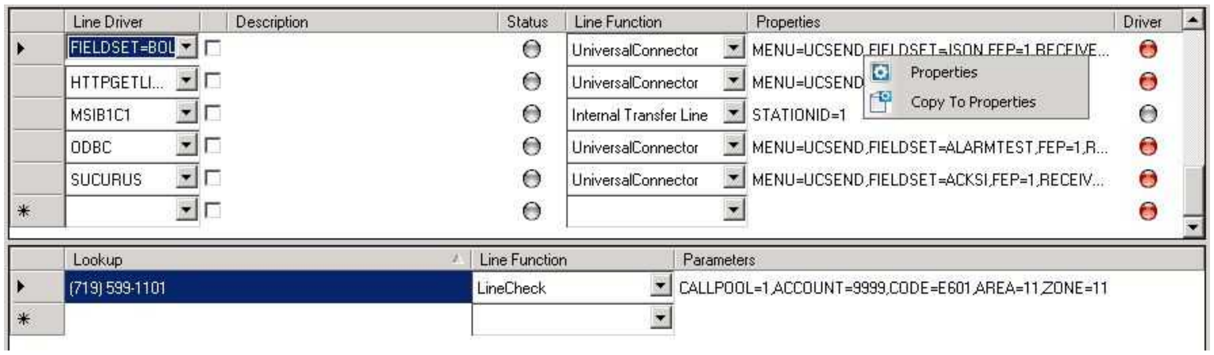


Line Driver	Description	Status	Line Function	Properties	Driver
BOLDSOS			AutoText		
CDYNE			UniversalConnector		
CLICKATELL			AutoText		
COM9				MENU=AUTO	
DXXXB2C1				MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	
DXXXB2C2				MENU=NATIONAL	

Lookup	Line Function	
(719) 599-1101	LineCheck	=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

Establish Properties

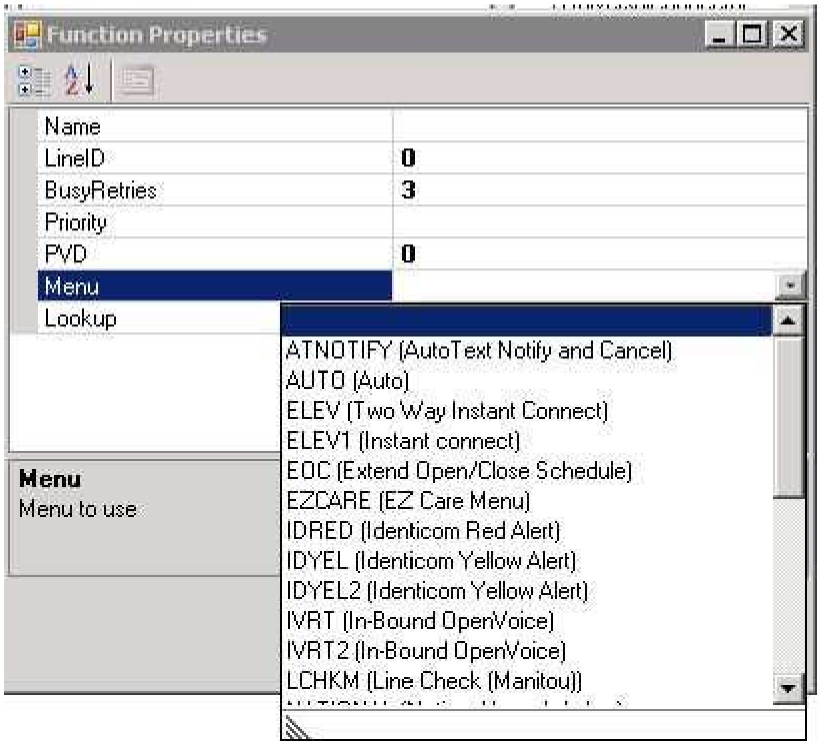
1. To enter property information for a Line Driver, select the Driver you want. Then, right-click in the Properties column and select Properties from the menu that displays. You also have the option of copying properties from an already existing Line Driver by selecting the "Copy to Properties" option as shown in the following screenshot:



Line Driver	Description	Status	Line Function	Properties	Driver
FIELDSET=BOL			UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	
HTTPGETLI...			UniversalConnector	MENU=UCSEND	
MSIB1C1			Internal Transfer Line	STATIONID=1	
ODBC			UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	
SUCURUS			UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	

Lookup	Line Function	Parameters
(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

Result: the Function Properties window displays as shown in the following screenshot:



Name	
LineID	0
BusyRetries	3
Priority	
PVD	0
Menu	
Lookup	

ATNOTIFY (AutoText Notify and Cancel)
AUTO (Auto)
ELEV (Two Way Instant Connect)
ELEV1 (Instant connect)
EOC (Extend Open/Close Schedule)
EZCARE (EZ Care Menu)
IDRED (Identicom Red Alert)
IDYEL (Identicom Yellow Alert)
IDYEL2 (Identicom Yellow Alert)
IVRT (In-Bound OpenVoice)
IVRT2 (In-Bound OpenVoice)
LCHKM (Line Check (Manitou))

Note: the function list items that display on the Function Properties menu depend on the Line Driver you selected.

2. Enter the values you want in the right column to correspond to the function list on the left. Some functions have dropdown menus (as shown in the screenshot above). To access a dropdown menu, click at the edge of the right column until a down arrow displays. Click the arrow, and scroll through the list to select the item you want.
3. Click "**OK**" when finished, and then click "**Save**".

File

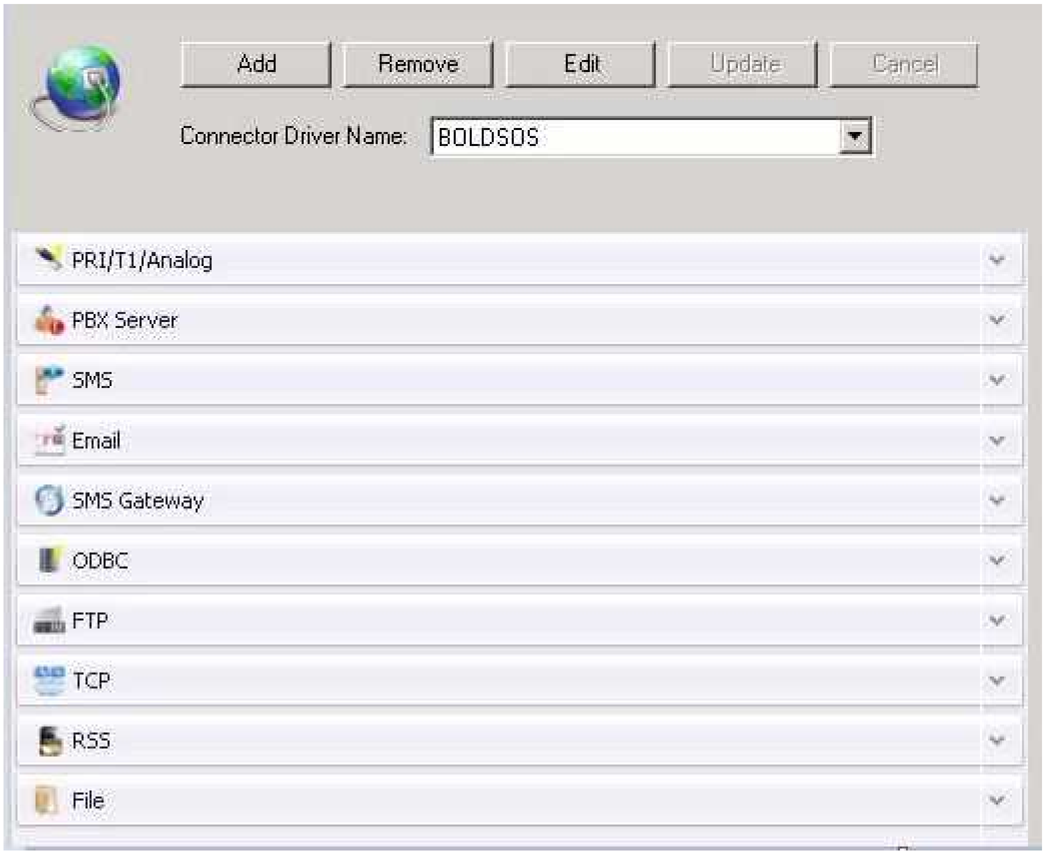
Adding a File Connector

The File Connector allows you to to look up simple files by prefix and extension.

Perform the following steps to add a File Connector:

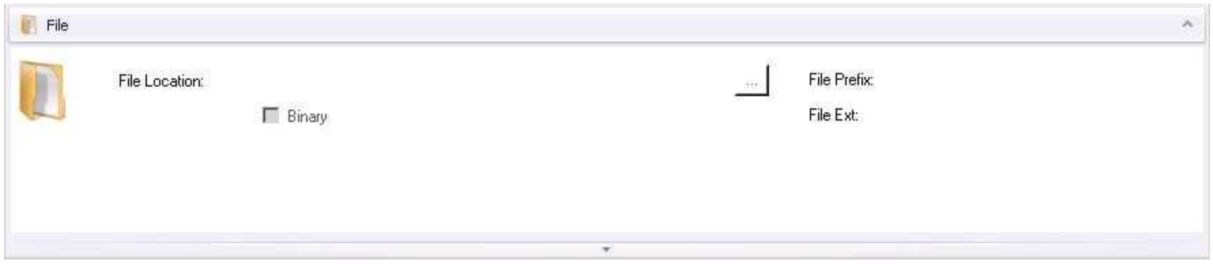
1. Click "Connector" under the UniversalConnector section of the Node Tree.

Result: the following window displays:

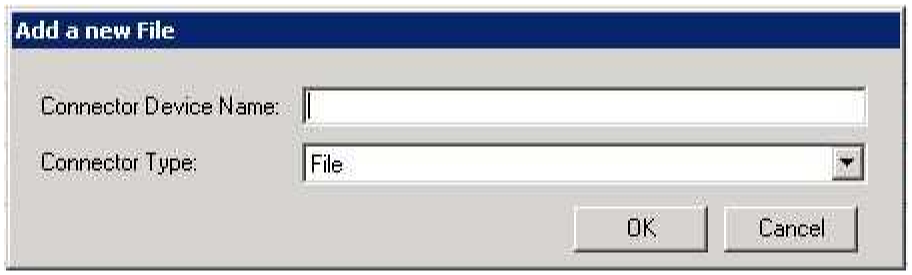


Connector Driver Name: BOLDSOS

2. Click the down arrow to expand the File area of the window as shown in the following screenshot:



3. Click "**Add**".
Result: the "Add a new File" dialog displays as shown in the following screenshot:



Connector Type: File

4. Enter a device name into the "Connector Device Name:" field.
5. Confirm that the "Connector Type:" field displays as "File". If it does not, select "File" from the dropdown list.
6. Click "**OK**".

Result: the "Add a new File" window closes and the fields on the File area of the window now display as populated with the default settings.

Note: you may need to modify the default settings depending on the hardware components you have installed. Use the dropdown arrows to access additional selections available for each field.

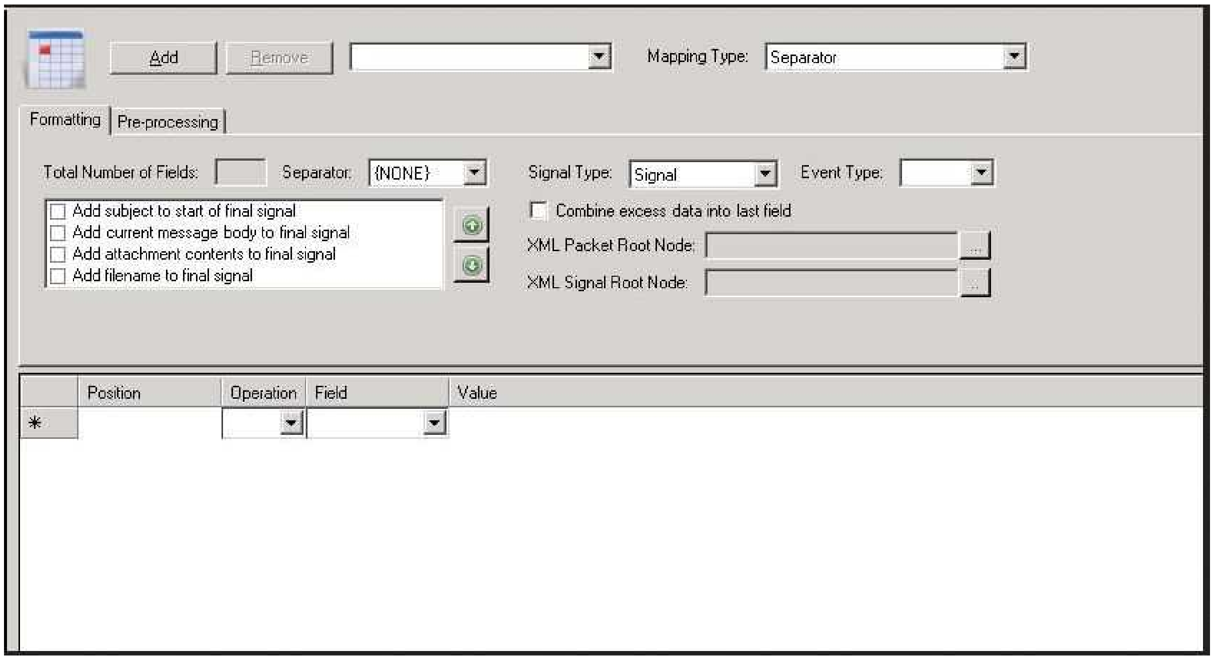
Data Mapping

In order for the communication to be properly identified and submitted to Manitou, you must define the correct data mapping in the Data Mapping form.

Perform the following steps to define the correct data mapping:

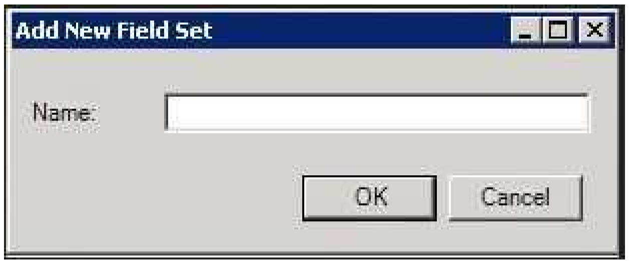
1. Click "Data Mapping" under UniversalConnector on the Node Tree.

Result: the Data Mapping form displays as shown in the following screenshot:

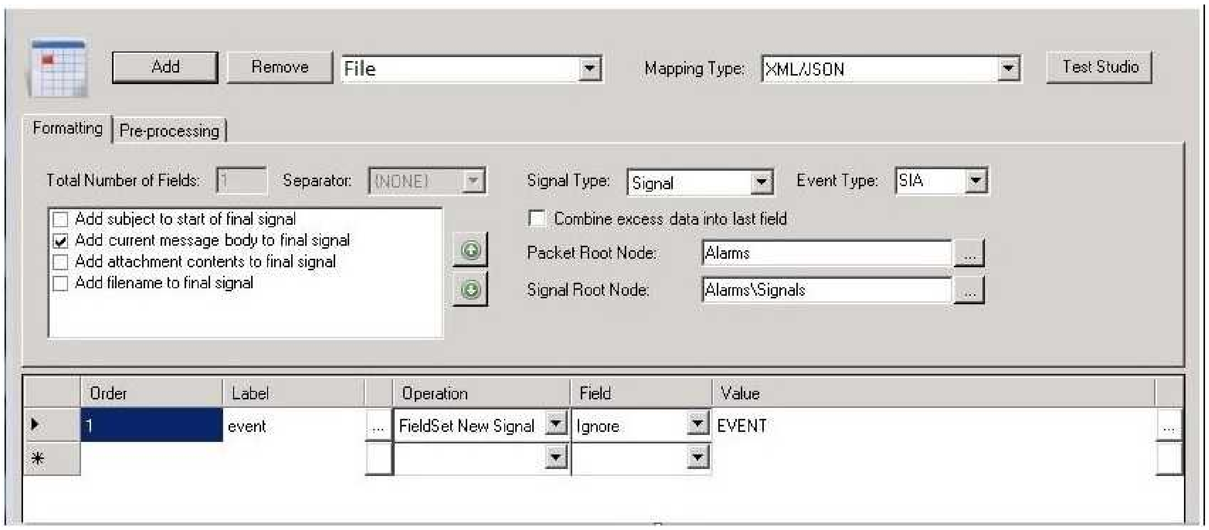


2. Select "File" from the dropdown menu directly to the right of the Remove button.
3. Click "**Add**".

Result: the "Add New Field Set" dialog displays as shown in the following screenshot:



4. Enter a name into the "Name:" field, and then click "**OK**".
Result: the "Add New Field Set" window closes and the system returns you to the Data Mapping form as shown in the following screenshot:



File

Mapping Type: XML/JSON

Total Number of Fields: 1 Separator: (NONE)

Signal Type: Signal Event Type: SIA

☐ Add subject to start of final signal
☑ Add current message body to final signal
☐ Add attachment contents to final signal
☐ Add filename to final signal

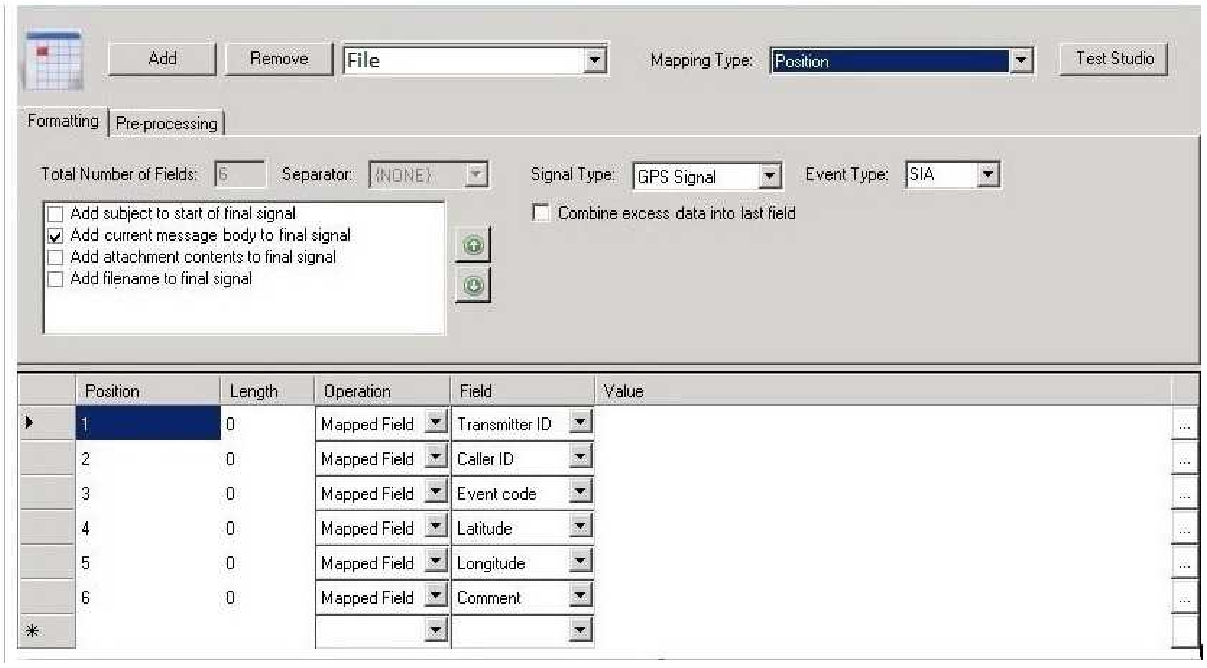
☐ Combine excess data into last field

Packet Root Node: Alarms

Signal Root Node: Alarms\Signals

Order	Label	Operation	Field	Value
1	event	FieldSet New Signal	Ignore	EVENT

5. Select an option from the "Mapping Type:" dropdown menu.
Note: your options are: Separator, Position, Label/Separator, and XML/JSON.
6. Select an option from the "Separator:" dropdown menu.
7. Select an option from the "Signal Type:" dropdown menu.
Note: your options are: Signal, GPS Signal, and Telemetry. For more information on these signal types, please refer to the document entitled "Field Descriptions".
8. Select an option from the "Event Type:" dropdown menu.
Note: your options are: SYS, SIA, and CID.
9. If you want to add part of the message to the final signal, select any of the four checkboxes.
10. If you want, select the "Combine excess data into the last field" checkbox.
11. In the Data Parameters section in the bottom portion of the screen, designate the order you want the information to be parsed into Manitou in the "Position" column.



File

Mapping Type: Position

Total Number of Fields: 6 Separator: (NONE) Signal Type: GPS Signal Event Type: SIA

- [ ] Add subject to start of final signal
- [x] Add current message body to final signal
- [ ] Add attachment contents to final signal
- [ ] Add filename to final signal

- [ ] Combine excess data into last field

Position	Length	Operation	Field	Value
1	0	Mapped Field	Transmitter ID	
2	0	Mapped Field	Caller ID	
3	0	Mapped Field	Event code	
4	0	Mapped Field	Latitude	
5	0	Mapped Field	Longitude	
6	0	Mapped Field	Comment	

Note: the column labels in the Data Parameters area change depending on the Mapping Type you selected.

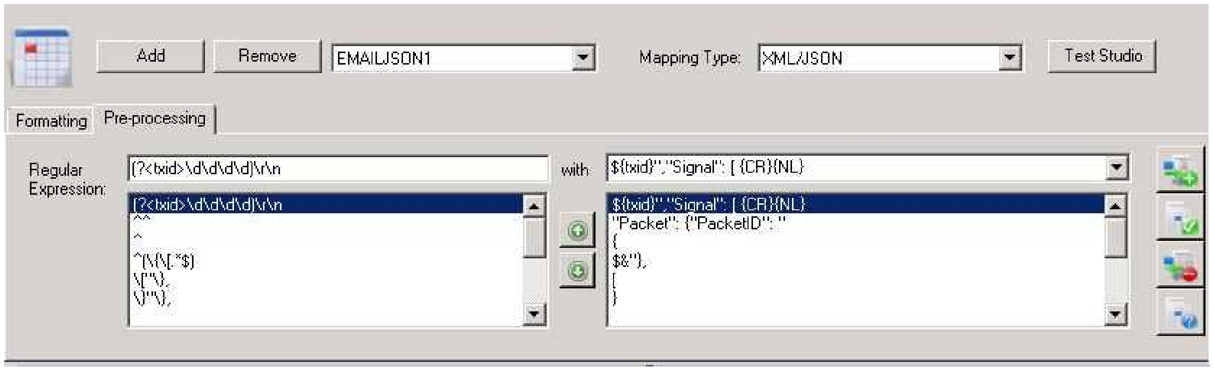
Data Mapping, Pre-processing

Defining Expressions in the Pre-processing Tab

The Pre-processing tab enables you to anticipate and prevent some common processing issues before they occur. For example, if a comma was designated as the separator on the Formatting tab, but there are instances where a semi-colon might come through the received signal as the separator, you can avoid this issue before it occurs.

Perform the following steps to use the Pre-processing tab to avoid common processing issues:

1. Click the Pre-processing tab on the Data Mapping form.
Result: the Pre-processing tab displays as shown in the following screenshot:



EMAILJSON1 Mapping Type: XML/JSON

Regular Expression: (?<txid>\d\d\d\d)\r\n with ${txid}","Signal": [{CR}{NL}

```
(?<txid>\d\d\d\d)\r\n
^^
^
^(\{\[.*$)
\["\},
\}"\},
```

```
${txid}","Signal": [ {CR}{NL}
"Packet": {"PacketID": "
{
$&"},
[
}
```

2. In the "Regular Expression:" field, enter the symbol or letters used as a common expression in the signals you receive.
3. Enter the interpretation option you want for the common expression in the right-hand field.

The buttons on the right-hand side of the window perform the following functions:

Add regular expression

Update regular expression

Remove regular expression

Test Studio

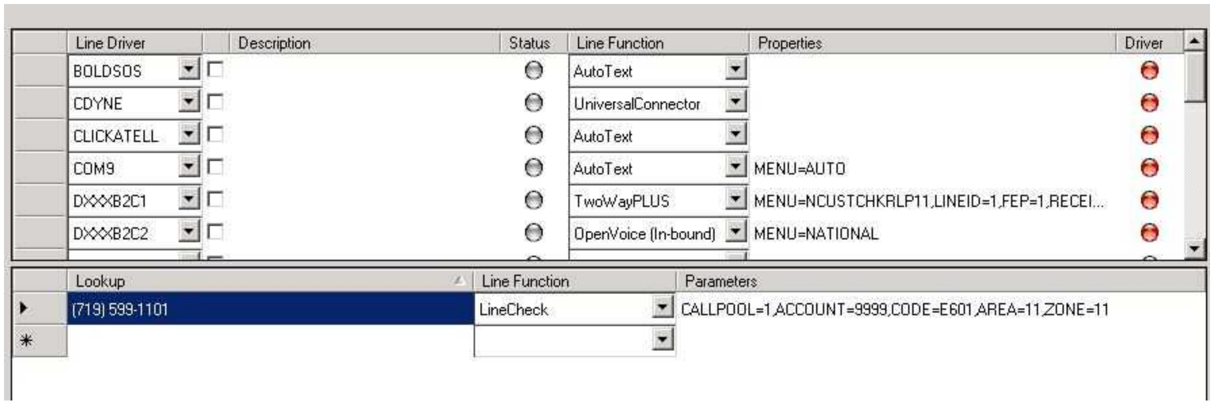
Line Drivers

Once you have created a File Connector and mapped it, you must format the Line Drivers.

Note: the FEP and associated Receivers must be correctly installed and configured prior to formatting Line Drivers. To check the installation of the FEPs and Receivers, please refer to the appropriate Supervisor Workstation manual for your version of Manitou.

Perform the following steps to format the Line Drivers:

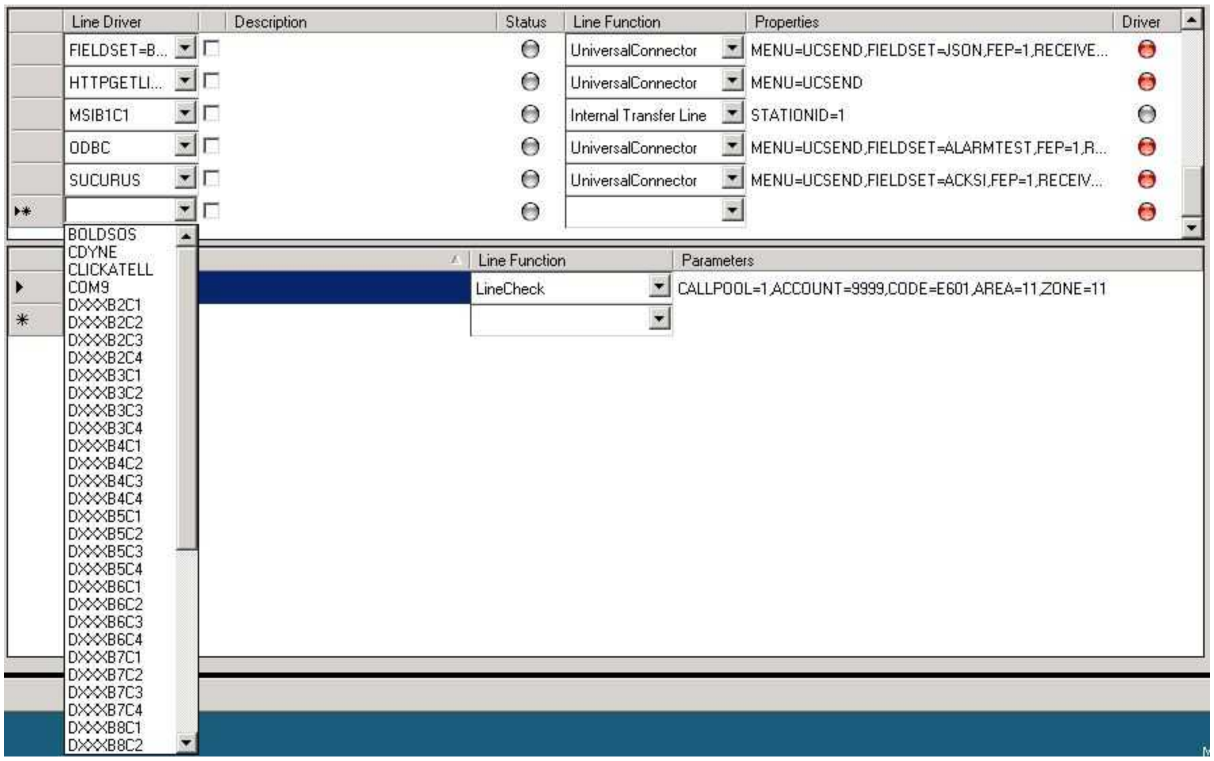
1. Click "Line Drivers" under Administration in the Node Tree.
Result: the Line Drivers form displays shown in the following screenshot:



Line Driver		Description	Status	Line Function	Properties	Driver
BOLDSOS	☐		●	AutoText		🔴
CDYNE	☐		●	UniversalConnector		🔴
CLICKATELL	☐		●	AutoText		🔴
COM9	☐		●	AutoText	MENU=AUTO	🔴
DXXXB2C1	☐		●	TwoWayPLUS	MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	🔴
DXXXB2C2	☐		●	OpenVoice (In-bound)	MENU=NATIONAL	🔴

	Lookup	Line Function	Parameters
▸	(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*			

2. In the "Line Driver" column, select the line driver from the pulldown list in the next open row (indicated with an asterisk) as shown in the following screenshot:

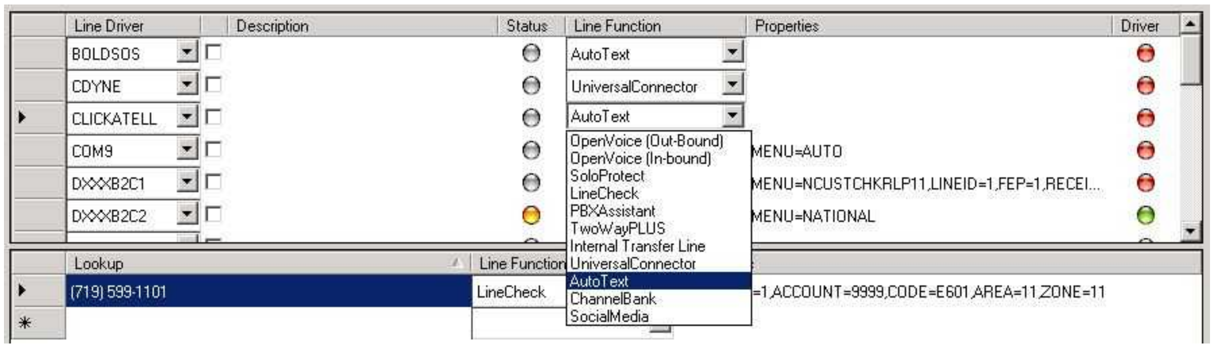


	Line Driver		Description	Status	Line Function	Properties	Driver
	FIELDSET=B...	☐		●	UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	🔴
	HTTPGETLI...	☐		●	UniversalConnector	MENU=UCSEND	🔴
	MSIB1C1	☐		●	Internal Transfer Line	STATIONID=1	●
	ODBC	☐		●	UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	🔴
	SUCURUS	☐		●	UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	🔴
▸*		☐		●			🔴

BOLDSOS
CDYNE
CLICKATELL
COM9
DXXXB2C1
DXXXB2C2
DXXXB2C3
DXXXB2C4
DXXXB3C1
DXXXB3C2
DXXXB3C3
DXXXB3C4
DXXXB4C1
DXXXB4C2
DXXXB4C3
DXXXB4C4
DXXXB5C1
DXXXB5C2
DXXXB5C3
DXXXB5C4
DXXXB6C1
DXXXB6C2
DXXXB6C3
DXXXB6C4
DXXXB7C1
DXXXB7C2
DXXXB7C3
DXXXB7C4
DXXXB8C1
DXXXB8C2

	Line Function	Parameters
▸	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11
*		

3. If you want the Line Driver to be designated as active, select the checkbox to the right of the driver. If you do not want it to be active, leave the checkbox deselected.
4. Enter a description into the "Description" field.
5. Select the type of "Line Function" you want from the pulldown menu as shown in the following screenshot:

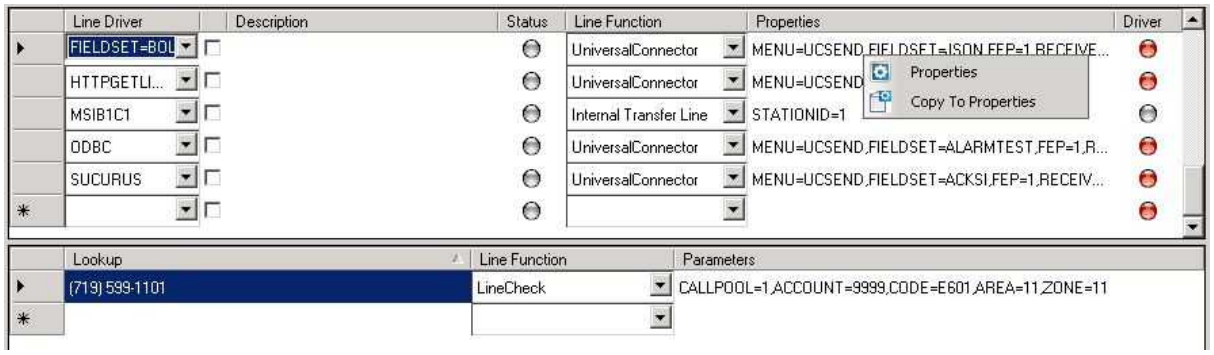


Line Driver	Description	Status	Line Function	Properties	Driver
BOLDSOS			AutoText		
CDYNE			UniversalConnector		
CLICKATELL			AutoText		
COM9				MENU=AUTO	
DXXXB2C1				MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	
DXXXB2C2				MENU=NATIONAL	

Lookup	Line Function	
(719) 599-1101	LineCheck	=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

Establish Properties

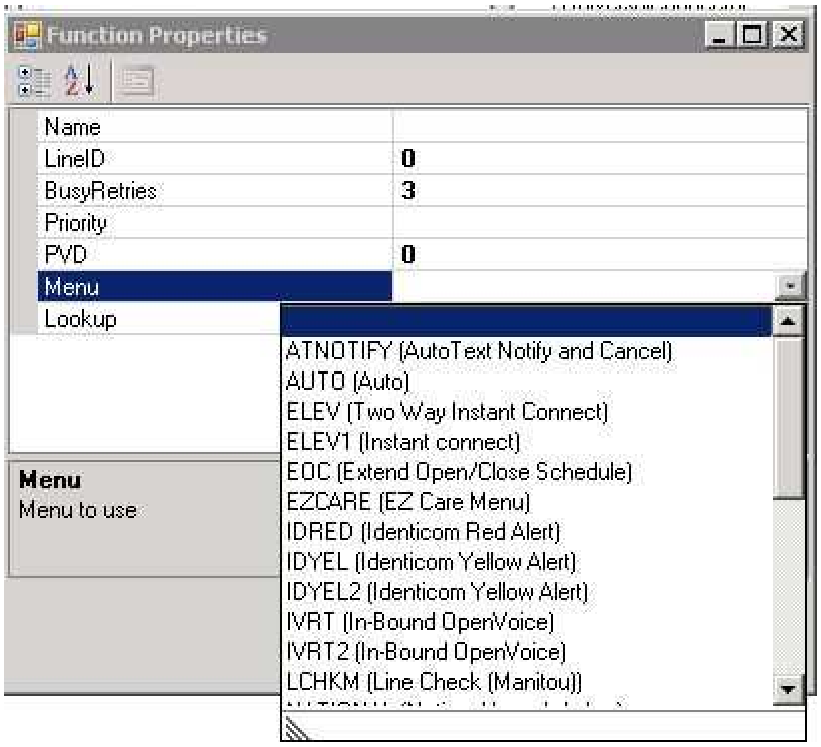
1. To enter property information for a Line Driver, select the Driver you want. Then, right-click in the "Properties" column and select Properties from the menu that displays. You also have the option of copying properties from an already existing Line Driver by selecting the "Copy to Properties" option as shown in the following screenshot:



Line Driver	Description	Status	Line Function	Properties	Driver
FIELDSET=BOL			UniversalConnector	MENU=UCSEND,FIELDSET=JSON,FEP=1,RECEIVE...	
HTTPGETLI...			UniversalConnector	MENU=UCSEND	
MSIB1C1			Internal Transfer Line	STATIONID=1	
ODBC			UniversalConnector	MENU=UCSEND,FIELDSET=ALARMTEST,FEP=1,R...	
SUCURUS			UniversalConnector	MENU=UCSEND,FIELDSET=ACKSI,FEP=1,RECEIV...	

Lookup	Line Function	Parameters
(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

Result: the Function Properties window displays as shown in the following screenshot:



Name	
LineID	0
BusyRetries	3
Priority	
PVD	0
Menu	
Lookup	

Menu
Menu to use

ATNOTIFY (AutoText Notify and Cancel)
AUTO (Auto)
ELEV (Two Way Instant Connect)
ELEV1 (Instant connect)
EOC (Extend Open/Close Schedule)
EZCARE (EZ Care Menu)
IDRED (Identicom Red Alert)
IDYEL (Identicom Yellow Alert)
IDYEL2 (Identicom Yellow Alert)
IVRT (In-Bound OpenVoice)
IVRT2 (In-Bound OpenVoice)
LCHKM (Line Check (Manitou))

Note: the function list items that display on the Function Properties menu depend on the Line Driver you selected.

2. Enter the values you want in the right column to correspond to the function list on the left. Some functions have dropdown menus (as shown in the screenshot above). To access a dropdown menu, click at the edge of the right column until a down arrow displays. Click the arrow, and scroll through the list to select the item you want.
3. Click "**OK**" when finished, and then click "**Save**".

PBX Assistant

The MediaGateway 2 is compatible with any PBX system. Universal compatibility is achieved through the use of standard analog lines, digital T1 lines, and digital ISDN PRI lines. In essence, the MediaGateway 2 becomes a specialized switch that works alongside your existing PBX system.

Use the PBX Assistant to accomplish any of the following tasks:

- Place incoming callers into a call queue
- Play hold messages to encourage callers to stay on the line
- Route incoming calls and queued callers to available Operators

How PBX Assistant Works

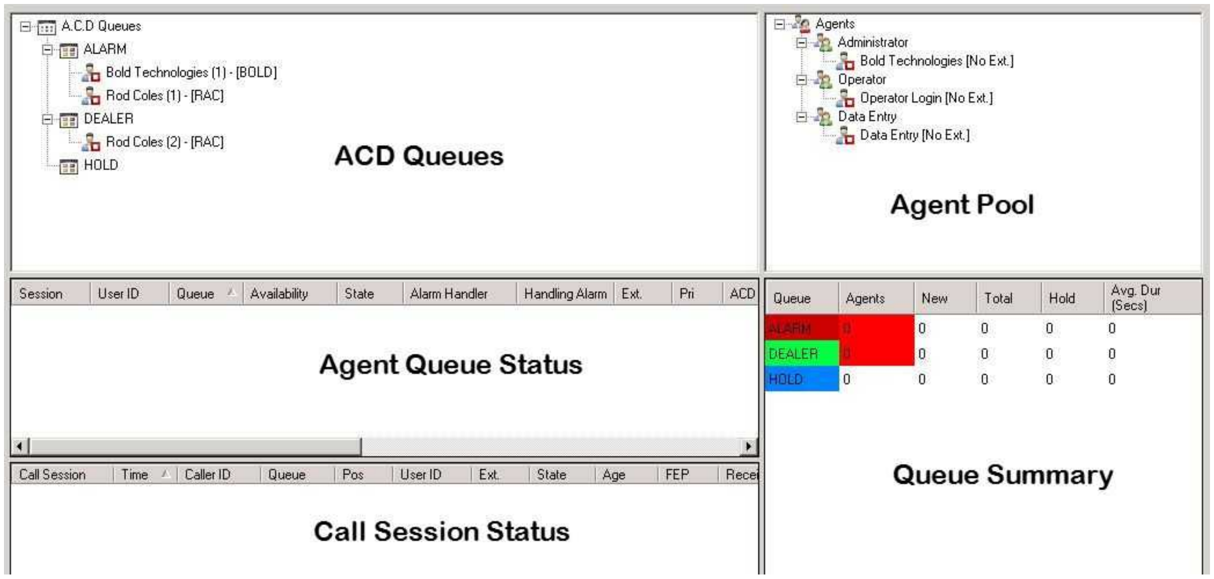
Automatic Call Distribution (ACD) is a method used for handling incoming calls. The PBX Assistant uses ACD to recognize and answer incoming calls by searching its database for call routing instructions using the ACD Queue menu. It then sends the call to an Operator according to the instructions for that call. The MediaGateway 2 can organize calls coming into the system into queues for handling by Operators. Calls can be routed through a variety of different queue types (for e.g., skills-based or round robin).

For example, a Customer accidentally trips his alarm and calls into the Central Station to cancel the alarm. After dialing in, the PBX Assistant prompts the Customer as to whether he would like Customer Service or Alarm Handling. The Customer chooses Alarm Handling, and the system identifies the caller by his caller ID. However, four other Customers are in the call queue in front of him because all the Operators who normally handle alarms are assisting other Customers. The system then notifies the Customer that Operators will be available shortly, and keeps him on hold while informing him of his expected wait time. When an Operator becomes available, the Customer is then routed to that Operator, where the alarm signal associated with his account is also directed.

PBX Assistant Main Form

The PBX Assistant form is divided into the following five panes:

- Main ACD Queue - you can view, edit, and add Operators to different queues
- Agent Queue Status - displays the status of each agent including his availability, extension, and the time and date of his last finished call
- Call Session Status - displays calls that are currently in the queue
- Agent Pool - allows you to add Operators to different queues
- Agent Status - shows how many Operators are in each ACD Queue, and the specific calls they are currently handling



- A.C.D Queues
  - ALARM
    - Bold Technologies (1) - [BOLD]
    - Rod Coles (1) - [RAC]
  - DEALER
    - Rod Coles (2) - [RAC]
  - HOLD

- Agents
  - Administrator
    - Bold Technologies [No Ext.]
  - Operator
    - Operator Login [No Ext.]
  - Data Entry
    - Data Entry [No Ext.]

Session	User ID	Queue	Availability	State	Alarm Handler	Handling Alarm	Ext.	Pri	ACD

Queue	Agents	New	Total	Hold	Avg. Dur (Secs)
ALARM	0	0	0	0	0
DEALER	0	0	0	0	0
HOLD	0	0	0	0	0

Call Session	Time	Caller ID	Queue	Pos	User ID	Ext.	State	Age	FEP	Recei

PBX Assistant form

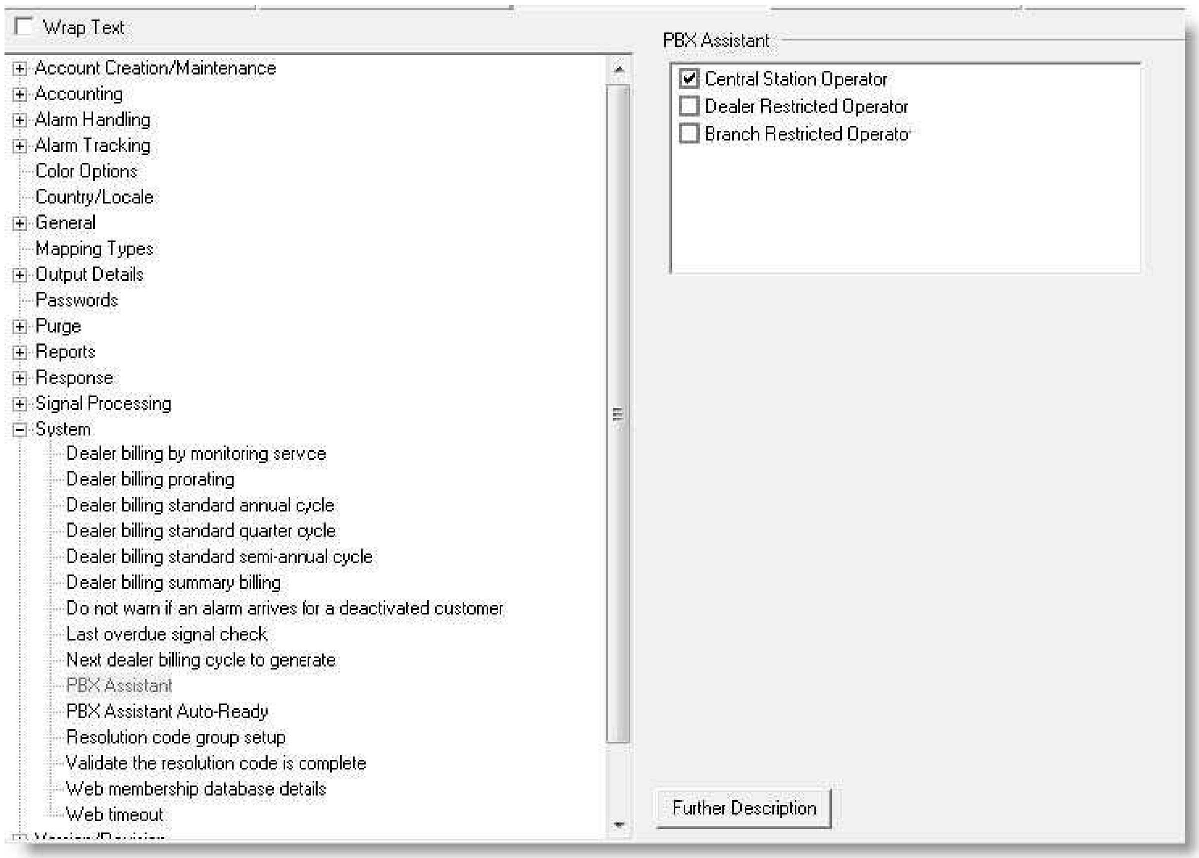
Setup

Before using the PBX Assistant, you must first set it up in the Manitou Supervisor Workstation, and then in the MediaGateway 2.

Manitou Setup

Perform the following steps to set up the PBX Assistant in Manitou:

1. Confirm that you are licensed for the PBX Assistant by navigating to the MediaGateway 2 "Options" menu, and then clicking "**Licensing**".
2. Open the Manitou Supervisor Workstation.
3. Navigate to the "Tools" menu, and then select "Options".
4. Expand "System" in the Navigation Tree, and then select "PBX Assistant" as shown in the following screenshot:



Wrap Text

- Account Creation/Maintenance
- Accounting
- Alarm Handling
- Alarm Tracking
- Color Options
- Country/Locale
- General
- Mapping Types
- Output Details
- Passwords
- Purge
- Reports
- Response
- Signal Processing
- System
  - Dealer billing by monitoring servce
  - Dealer billing prorating
  - Dealer billing standard annual cycle
  - Dealer billing standard quarter cycle
  - Dealer billing standard semi-annual cycle
  - Dealer billing summary billing
  - Do not warn if an alarm arrives for a deactivated customer
  - Last overdue signal check
  - Next dealer billing cycle to generate
  - PBX Assistant
  - PBX Assistant Auto-Ready
  - Resolution code group setup
  - Validate the resolution code is complete
  - Web membership database details
  - Web timeout

PBX Assistant

- [✓] Central Station Operator
- [ ] Dealer Restricted Operator
- [ ] Branch Restricted Operato

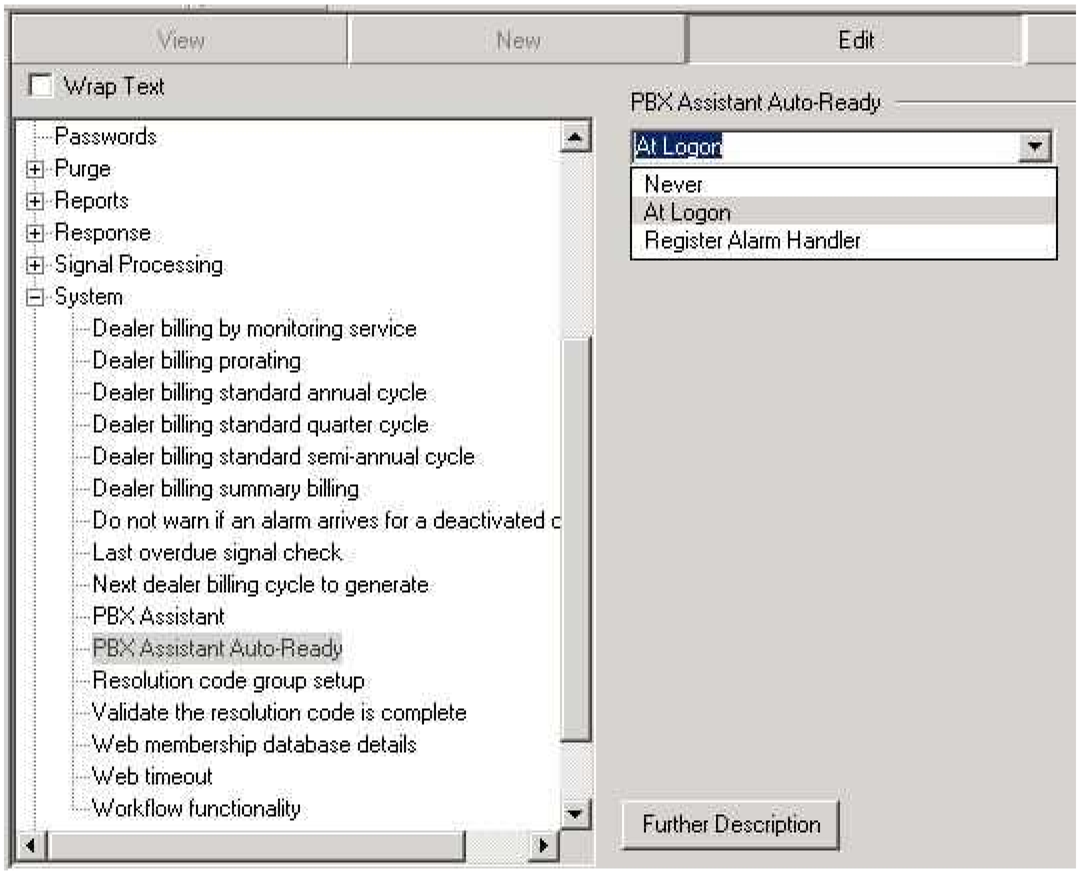
Further Description

Manitou Supervisor System Options

5. Click "**Edit**".
6. Select the type of Operator to which the PBX Assistant will be available when logging onto Manitou.

Note: most often, this is the "Central Station Operator". Your other options are "Dealer Restricted Operator" and "Branch Restricted Operator". You can select multiple options.

7. Next, select "PBX Assistant Auto-Ready" from the Navigation Tree.
8. Select an option from the "PBX Assistant Auto-Ready" dropdown menu as shown in the following screenshot:



- Passwords
- Purge
- Reports
- Response
- Signal Processing
- System
  - Dealer billing by monitoring service
  - Dealer billing prorating
  - Dealer billing standard annual cycle
  - Dealer billing standard quarter cycle
  - Dealer billing standard semi-annual cycle
  - Dealer billing summary billing
  - Do not warn if an alarm arrives for a deactivated c
  - Last overdue signal check
  - Next dealer billing cycle to generate
  - PBX Assistant
  - PBX Assistant Auto-Ready
  - Resolution code group setup
  - Validate the resolution code is complete
  - Web membership database details
  - Web timeout
  - Workflow functionality

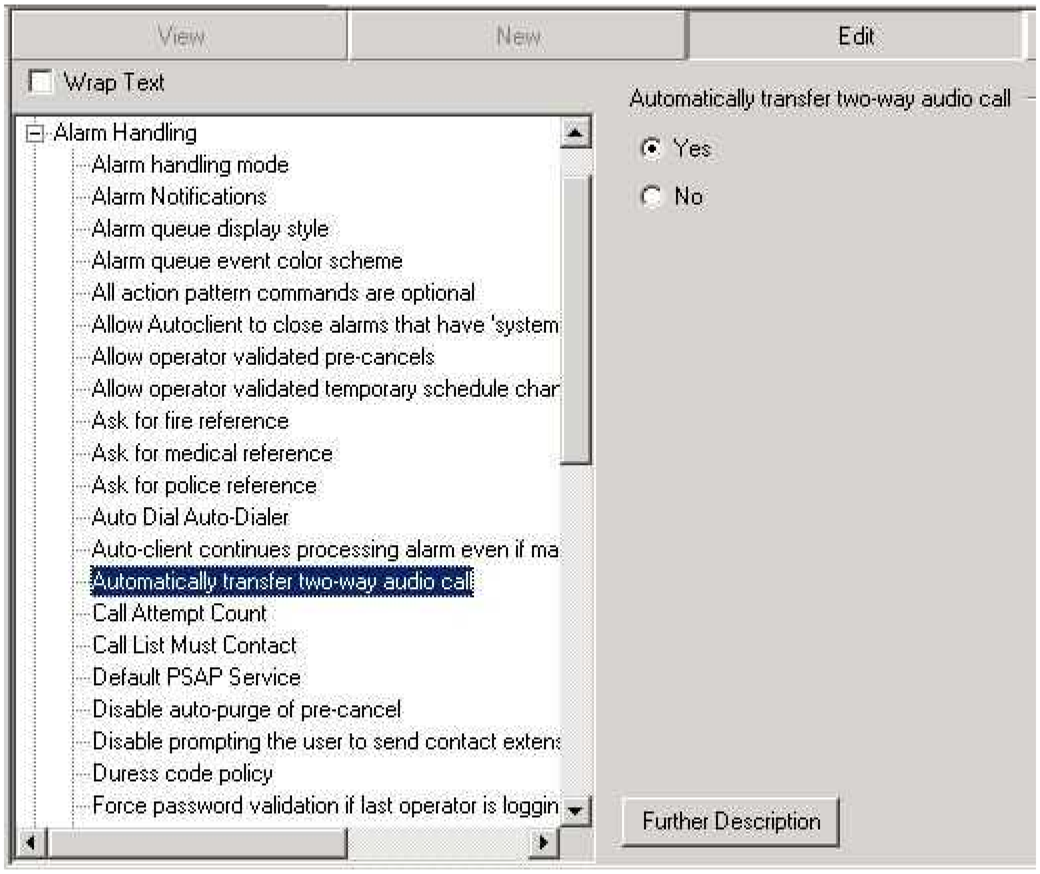
PBX Assistant Auto-Ready: At Logon

- Never
- At Logon
- Register Alarm Handler

Note: this option tells the system when an Operator is available for the PBX Assistant. You have the following three options:

- Never: users have to manually make themselves available
- At Logon: Operators become available when they logged into Manitou
- Register Alarm Handler: the Operator becomes available when the agent enters alarm handling mode

10. Collapse the System section of the Navigation Tree, and then expand the Alarm Handling Section.
11. Select "Automatically transfer two-way audio calls" as shown in the following screenshot:



Alarm Handling
- Alarm handling mode
- Alarm Notifications
- Alarm queue display style
- Alarm queue event color scheme
- All action pattern commands are optional
- Allow Autoclient to close alarms that have 'system
- Allow operator validated pre-cancels
- Allow operator validated temporary schedule char
- Ask for fire reference
- Ask for medical reference
- Ask for police reference
- Auto Dial Auto-Dialer
- Auto-client continues processing alarm even if ma
- Automatically transfer two-way audio call
- Call Attempt Count
- Call List Must Contact
- Default PSAP Service
- Disable auto-purge of pre-cancel
- Disable prompting the user to send contact extens
- Duress code policy
- Force password validation if last operator is loggin

Automatically transfer two-way audio call

[✓ Yes] No

12.Select an option for "Automatically transfer two-way audio call".
Note: you have the following two options:

- No - calls route to the PBX Assistant, and Operators can choose whether or not to take them
- Yes - the PBX Assistant automatically transfers any two-way audio calls associated with an alarm to the Operator's extension

13.Click "**Save**".

Adding Users and User Groups in Manitou

In order for Operators and Supervisors to be able to use the PBX Assistant, you must first include them in a Manitou User Group.

Perform the following steps to add users to a Manitou User Group:

1. Open the Supervisor Workstation.
2. Navigate to the Maintenance menu, and select "Users".
3. Access the record for the user you want, and then select an option from the "User

Group" dropdown menu. The user is added to the User Group you select.
4. Enter the user's extension into the "Extension" field. This is the extension to which the PBX Assistant routes the calls. It is possible to enter multiple extensions, if you separate them by commas.

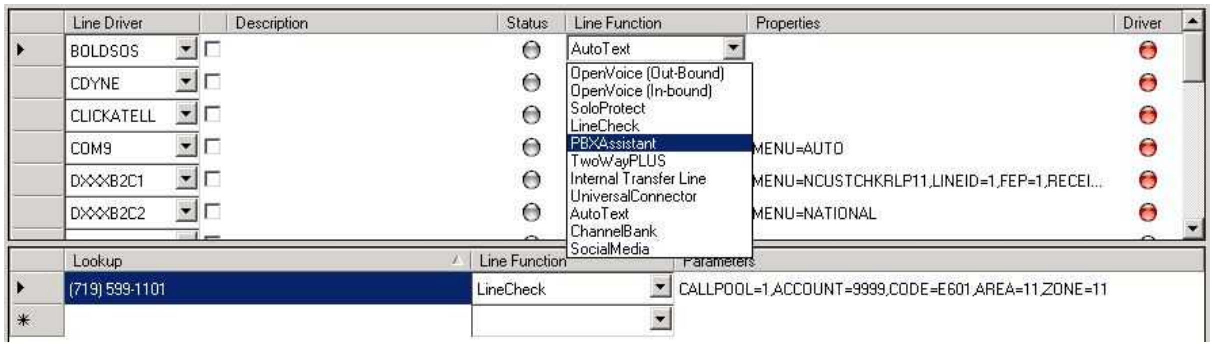
MediaGateway 2 Setup

Line Driver Configuration

After you complete the Manitou setup, you must configure the Line Drivers for the PBX Assistant.

Perform the following steps to configure the line drivers:

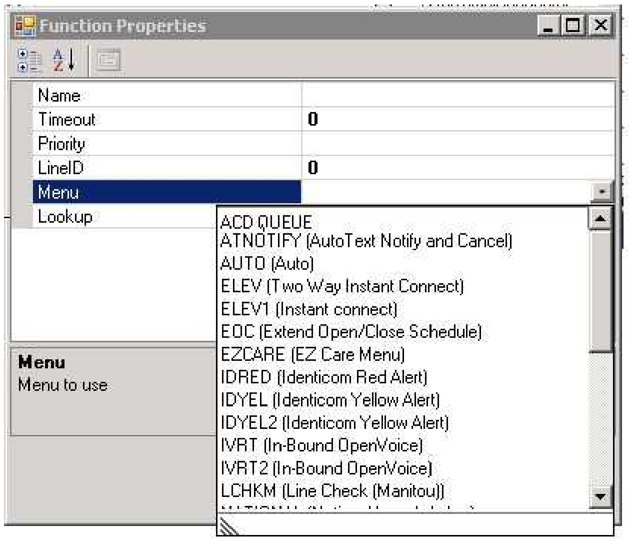
1. Open the MediaGateway 2.
2. Click "Line Drivers" from the Navigation Tree.
3. Select the Line Driver you want to associate with the PBX Assistant, and enter a Description if you want.
4. Select "PBX Assistant" from the Line Function dropdown menu as shown in the following screenshot:



Line Driver	Description	Status	Line Function	Properties	Driver
BOLDSOS			AutoText		
CDYNE					
CLICKATELL					
COM9				MENU=AUTO	
DXXXB2C1				MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	
DXXXB2C2				MENU=NATIONAL	

Lookup	Line Function	Parameters
(719) 599-1101	LineCheck	CALLPOOL=1,ACCOUNT=9999,CODE=E601,AREA=11,ZONE=11

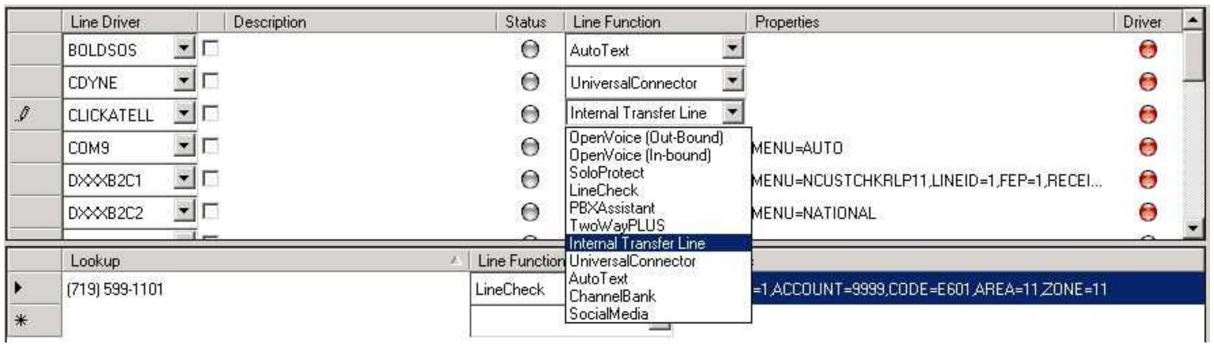
5. Right-click in the "Properties" area on the row for the Line Driver you associated with the PBX Assistant, and click at the right edge of the Menu row until a dropdown menu displays.
6. Select the "ACD Queue" menu as shown in the following screenshot:


Function Properties
Name
Timeout 0
Priority
LineID 0
Menu
Lookup
Menu
Menu to use
ACD QUEUE
ATNOTIFY (AutoText Notify and Cancel)
AUTO (Auto)
ELEV (Two Way Instant Connect)
ELEV1 (Instant connect)
EOC (Extend Open/Close Schedule)
EZCARE (EZ Care Menu)
IDRED (Identicom Red Alert)
IDYEL (Identicom Yellow Alert)
IDYEL2 (Identicom Yellow Alert)
IVRT (In-Bound OpenVoice)
IVRT2 (In-Bound OpenVoice)
LCHKM (Line Check (Manitou))

7. Click in the LineID Properties area (still on the Function Properties) window.
8. Enter the Line ID.

Note: this number must match the internal line transfer Line ID.

9. Next, select the Line Driver you want to associate with the PBX Assistant Transfer Line.
10. Select "Internal Transfer Line" for that row from the Line Function drop-down menu as shown in the following screenshot:



11. Right-click in the Properties area, and then enter the LineID on the Function Property window.

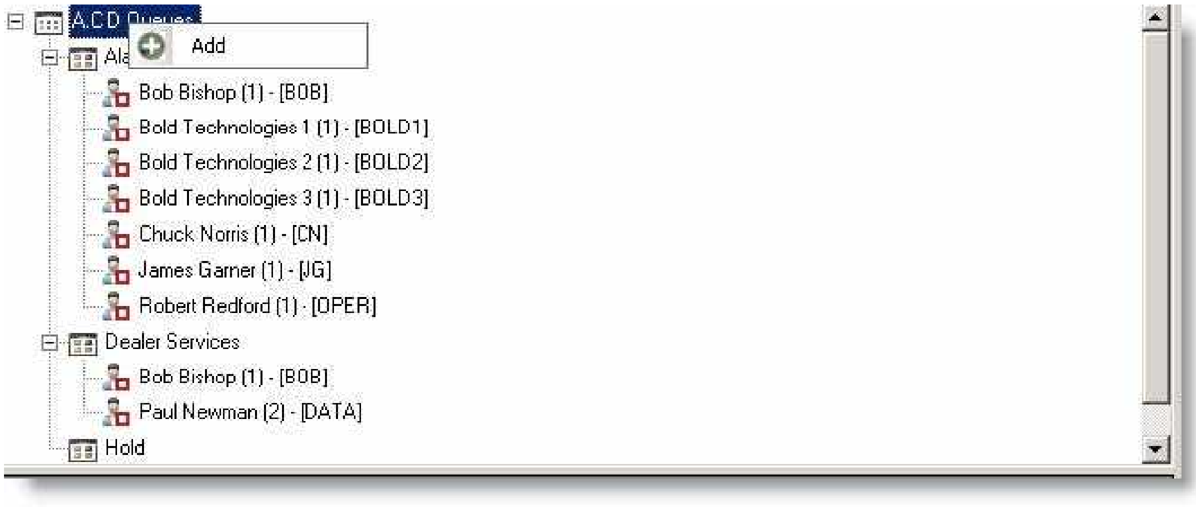
Note: this value must match the Line IDs you set up for the PBX Assistant Line Driver.

Setting up the PBX Assistant in the MediaGateway 2

Now that you have added and configured the necessary Line Drivers, you must create an ACD Queue in the MediaGateway 2. The following example creates an Alarm Handling Queue.

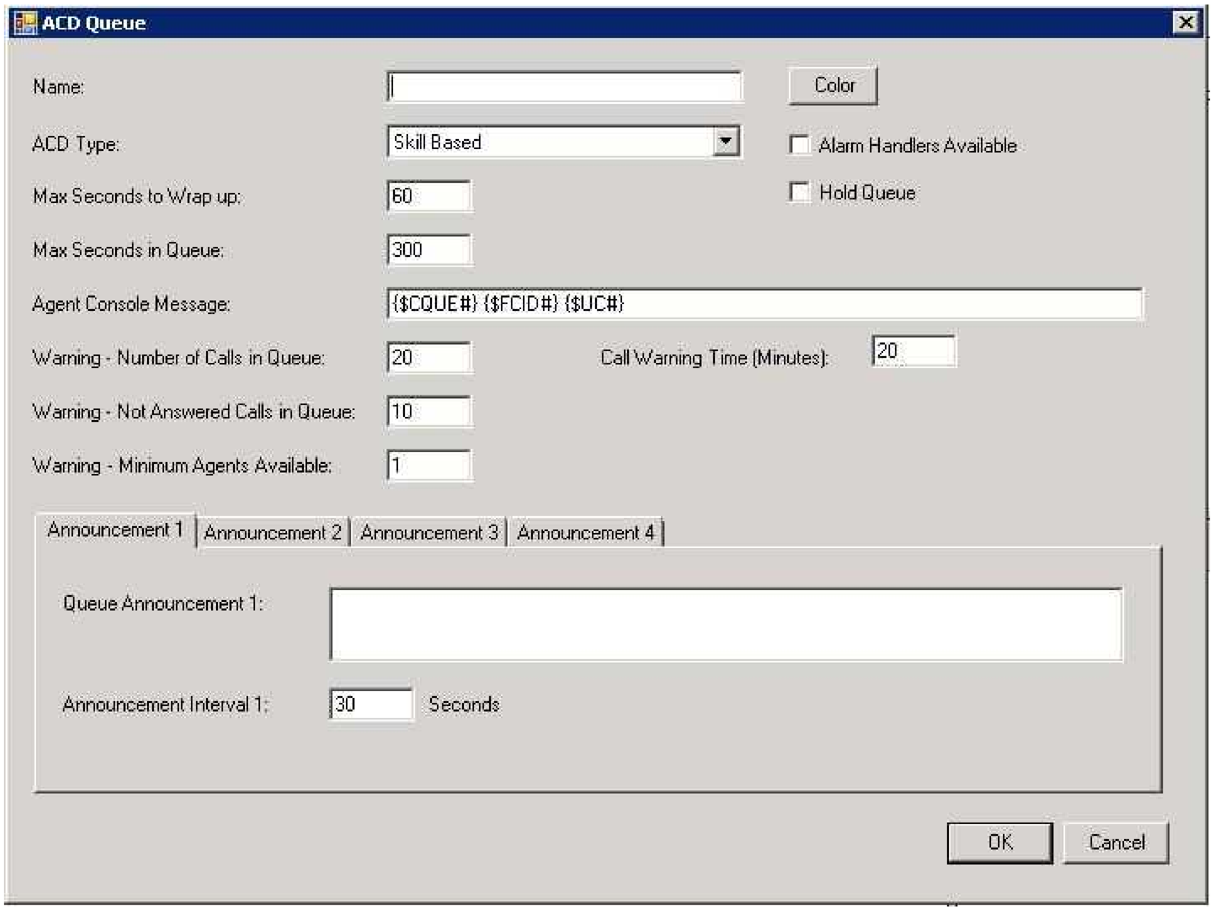
1. Select the ACD Queue menu on the PBX Assistant form in the MediaGateway 2. Then,

right-click and click "**Add**".



Add ACD Queue

Result: the ACD Queue form displays as shown in the following screenshot:



Name:

ACD Type: Skill Based

Alarm Handlers Available

Hold Queue

Max Seconds to Wrap up: 60

Max Seconds in Queue: 300

Agent Console Message: {$CQUE#} {$FCID#} {$UC#}

Warning - Number of Calls in Queue: 20

Call Warning Time (Minutes): 20

Warning - Not Answered Calls in Queue: 10

Warning - Minimum Agents Available: 1

Queue Announcement 1:

Announcement Interval 1: 30 Seconds

Creating an ACD Queue

Perform the following steps to create an ACD Queue:

1. From the "ACD Queue" form, enter a name for the queue (for e.g., "Alarm Handling").
2. Select a type from the "ACD Type:" dropdown menu.

Note: this is the routing mechanism the MediaGateway 2 uses to find an Operator. The following options are available:

- Skill Based - Use this type of routing to assign incoming calls to the most skilled Operator. For example, in an Alarm Handling ACD Queue, calls should be routed to Operators who routinely handle alarms rather than those who only perform data entry. Each Operator has a priority level associated with his skill set.
- First in List - With this type of routing, the first Operator on the list receives every call. If the first Operator is not available, the MediaGateway 2 routes the call to the next Operator.
- Round Robin - With Round Robin routing, each Operator receives calls in succession beginning with the last Operator to answer a call.
- Longest Idle - The Operator who has waited the longest since the last call is the first to receive the next call.
- Average Longest Idle - MediaGateway 2 calculates the Operator with the longest average idle time, and then routes the next call to him.
- Fewest Calls - MediaGateway 2 routes all future calls to the Operator who has answered the fewest calls.
- Shortest Average Talk Time - MediaGateway 2 routes all future calls to the Operator who has spent the least amount of time handling calls.

3. If the ACD Queue you are creating is an alarm handling queue, select the "Alarm Handlers Available" option.

Note: this option determines that if a caller in the queue is calling about an alarm currently being handled by an Operator, the MediaGateway 2 routes that call to that Operator. If the Operator has suspended or deferred the alarm after handling it, the MediaGateway 2 also routes the call to the Operator who is tracking the alarm. Normally, the MediaGateway 2 removes an Operator from handling the call queues when he is handling an alarm so he can finish handling it.

4. Select the the "Hold Queue" checkbox if you want to place on-hold calls into a special hold queue.

Note: hold queues are similar to other queues with the exception that the system does not attempt to find an available Operator when a caller is placed into the hold queue. Instead, the caller is returned to the original Operator once they are removed from being on hold.

5. Enter the "Max Seconds to Wrap Up:" time.

Note: this is the the maximum time you want to allocate to an Operator before he can receive another call. This time allows him to do any additional work after a call finishes.

6. Enter the "Max Seconds in Queue:" time.

Note: this is the maximum time the caller will remain in the call queue before exiting. You can set the ACD Queue menu to loop back out, and put the caller into another queue. You can also set it to ask the caller if he wants to remain on the line or leave a message. In a typical scenario, the caller gets added to the call queue, and is removed when an Operator answers the call and handles it successfully. However, if the caller remains in the queue and the "Max Seconds in Queue:" time limit is reached, the caller is removed from the hold queue.

7. Enter an "Agent Console Message:".

Note: this is the message the Operator sees displayed in the Manitou Operator Workstation when he receives a PBX Assistant call. The default settings are as follows:

- {$CQUE#} is the queue name (for e.g., Alarm Handling)
- {$FCID#} is the formatted caller ID
- {$UC#} is the contact name

For example, the Operator might see the following message in Manitou: "Alarm Handling (719) 555-1234 John Doe". Right-clicking in the "Agent Console Message:" field also displays the "Variables" dialog. You can change which variables display to an Operator. Not all variables can be used with all types of calls.

Warnings

Warnings are entered values that, when triggered, cause fields to display red in the call summary view. This visual warning is meant to alert Operators and Supervisors of special circumstances (for e.g., too many calls in the queue or too few Operators available).

Announcements

Announcements are outgoing messages played at various intervals while a caller is waiting in a queue. Like the Agent Console Message, you can right-click in the announcement field and add variables. Typically, Announcements use the following two variables: {$QWT#} and {$QPOS#}. These represent the caller's queue wait time and his queue position, respectively.

Adding Operators to Queues

1. Once you have set up the ACD Queue, click **"OK"** to return to the main PBX Assistant screen. The Alarm Handling Queue you just created now displays in the ACD Queues Navigation Tree.
2. Click on an Operator in the Agent Pool and drag the name to the ACD Queue in which you want the Operator to be. The system then prompts you to enter a priority level for the Operator. The priority level is used primarily for Skill Based routing, but can also be used to sequence Operators for other ACD Queue types, such as Round Robin routing.

Agent Status

- Operators who are online and available to take calls are denoted by a green check mark.
- Operators who are offline are denoted by a red square.
- Operators who are not ready to handle a call (i.e., they are already handling a call or finishing up alarm handling) have a red "No Entry" icon.

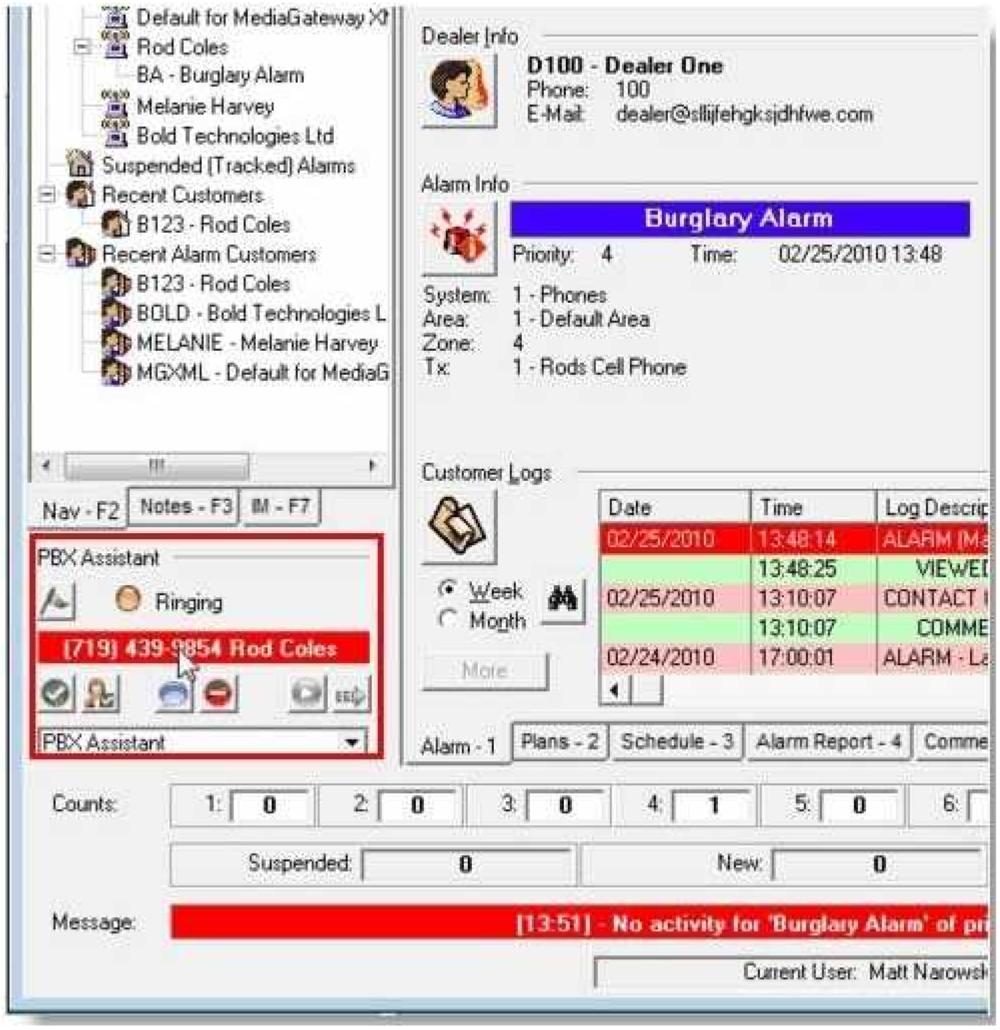
Agent Queue Status

The Agent Queue Status window displays all the queues and Operators and their different states of preparedness (i.e., Ready, Offline, or Not Ready). If an Operator is not logged into Manitou, he will not display in the Agent Queue Status window. If the queue is an alarm handling queue, the Alarm Handler and Handling Alarm columns display as highlighted.

- If the Alarm Handler column displays red, it indicates that an Operator is not in alarm handling mode (even though it is an alarm handling queue).
- A green Alarm Handler column indicates that an Operator is in alarm handling mode.
- If the Alarm Handling column displays yellow, it indicates that an Operator is already handling an alarm and is not receiving calls unless the call relates to the alarm he is handling.

PBX Assistant and Manitou

Once you have enabled the necessary options, the PBX Assistant displays to Operators in the Manitou alarm screen right under the main Navigation Tree as shown in the following screenshot:



Default for MediaGateway X
Rod Coles
BA - Burglary Alarm
Melanie Harvey
Bold Technologies Ltd
Suspended (Tracked) Alarms
Recent Customers
B123 - Rod Coles
Recent Alarm Customers
B123 - Rod Coles
BOLD - Bold Technologies L
MELANIE - Melanie Harvey
MGXML - Default for MediaG

PBX Assistant
Ringing
(719) 439-9854 Rod Coles
PBX Assistant

Dealer Info
D100 - Dealer One
Phone: 100
E-Mail: dealer@sllijfehgksjdhfwe.com

Alarm Info
Burglary Alarm
Priority: 4 Time: 02/25/2010 13:48
System: 1 - Phones
Area: 1 - Default Area
Zone: 4
Tx: 1 - Rods Cell Phone

Customer Logs

Date	Time	Log Descrip
02/25/2010	13:48:14	ALARM (Ma
	13:48:25	VIEWED
02/25/2010	13:10:07	CONTACT
	13:10:07	COMME
02/24/2010	17:00:01	ALARM - La

Counts: 1: 0 2: 0 3: 0 4: 1 5: 0 6:
Suspended: 0 New: 0
Message: [13:51] - No activity for 'Burglary Alarm' of pri
Current User: Matt Narowsk

Manitou Alarm Handling screen, PBX Assistant

Customer information displays in a single line in Manitou. You can modify how Customer information displays in Manitou in the set up for the ACD Queue.

Receiving a Call

When a call is received, the Operator can choose between any of the following actions:

- Accepting the the call
- Accepting the call and loading the Customer (**Note:** the MediaGateway 2 must identify the Customer before this can occur)
- Transferring the call to another Operator
- Putting the caller on hold
- Playing a message to the caller while he is on hold
- Rejecting the call
- Hanging up on the caller

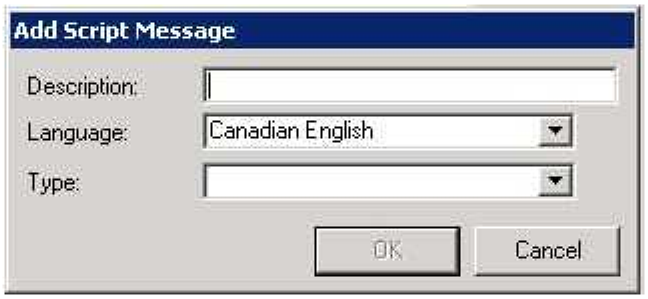
Setting up Scripts in the Manitou Supervisor Workstation

Operators can play scripts while a Customer is on hold. In order to use this option, you must first create the scripts you want in the Manitou Supervisor Workstation.

Perform the following steps to create scripts in the Supervisor Workstation:

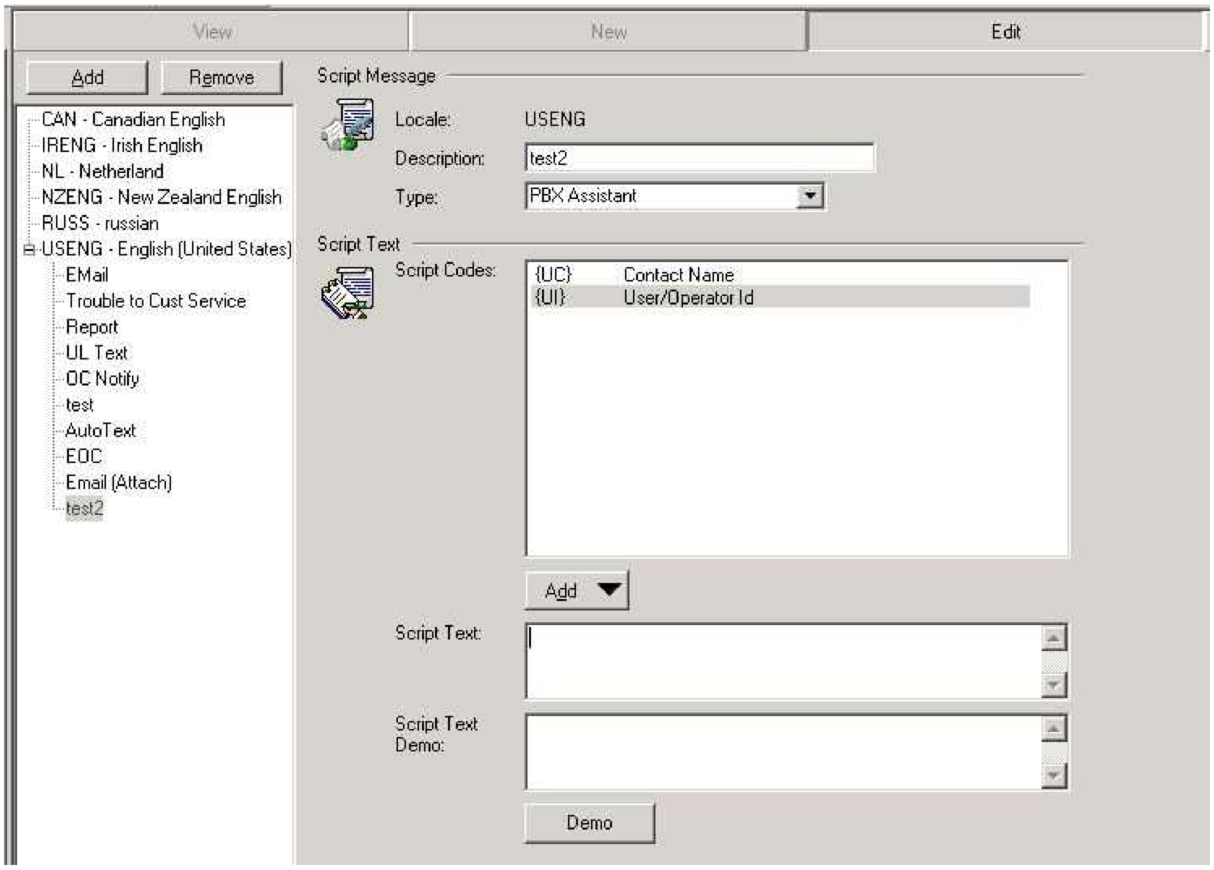
1. Open the Supervisor Workstation.
2. Navigate to the Maintenance menu, and then select "Script Messages".
3. Click "**Edit**", and then click "**Add**".

Result: the "Add Script Message" window displays as shown in the following screenshot:



4. Select a language for your script message from the "Language:" dropdown menu.
5. Select "PBX Assistant" from the "Type:" dropdown menu.
6. Click "**OK"**.

Result: the "Add Script Message" window closes and the "Script Message" window displays as shown in the following screenshot:



CAN - Canadian English
IRENG - Irish English
NL - Netherland
NZENG - New Zealand English
RUSS - russian
USENG - English (United States)
EMail
Trouble to Cust Service
Report
UL Text
OC Notify
test
AutoText
EOC
Email (Attach)
test2

Script Message

Locale: USENG
Description: test2
Type: PBX Assistant

Script Text

Script Codes:

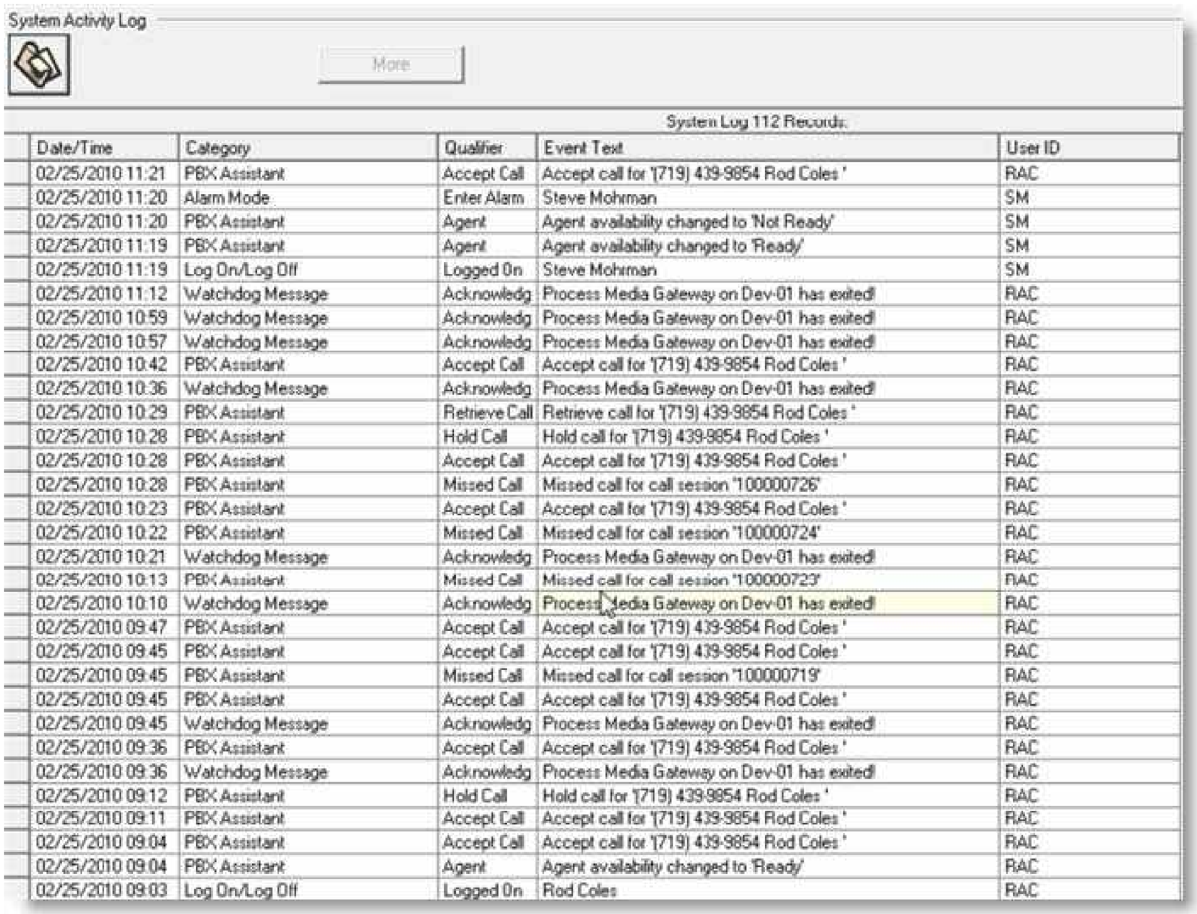
{UC}	Contact Name
{UI}	User/Operator Id

7. Double-click an option from the "Script Codes:" field if you want to include it in your script message.
8. Enter the text for the message you want to create.
9. If you want to preview how the message will play to the Customer, click "**Demo**".
10. Click "**Save**".

Now, whenever an Operator places a Customer on hold, he can select the script he wants from the "Script" dropdown menu on the Operator Workstation Alarm Handling window, and play it by pressing "**Play Script".**

System Log

The PBX Assistant displays in the Manitou System Log. The System Log displays call acceptance, status changes, hanging up, and missing calls, etc.



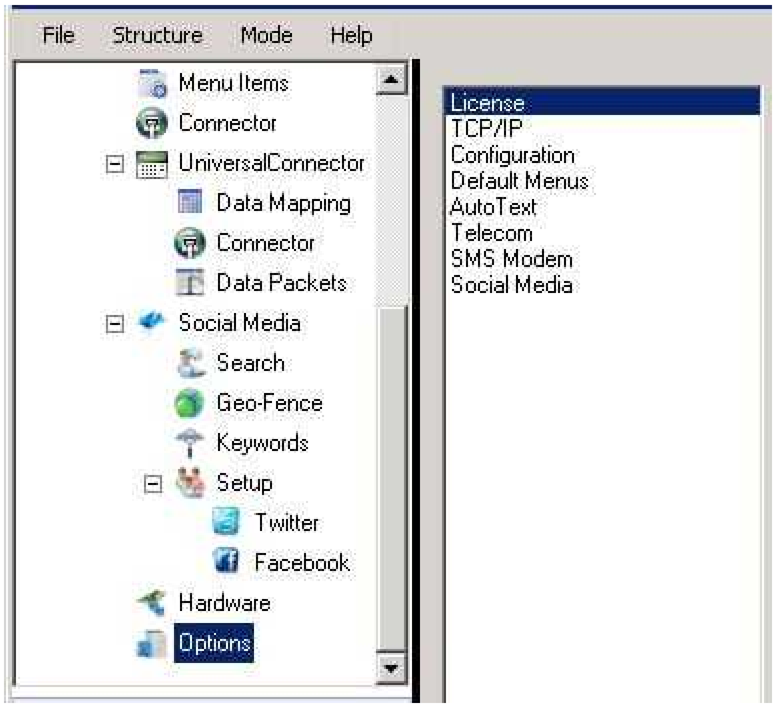
System Log 112 Records:

Date/Time	Category	Qualifier	Event Text	User ID
02/25/2010 11:21	PBX Assistant	Accept Call	Accept call for '(719) 439-9854 Rod Coles '	RAC
02/25/2010 11:20	Alarm Mode	Enter Alarm	Steve Mohrman	SM
02/25/2010 11:20	PBX Assistant	Agent	Agent availability changed to 'Not Ready'	SM
02/25/2010 11:19	PBX Assistant	Agent	Agent availability changed to 'Ready'	SM
02/25/2010 11:19	Log On/Log Off	Logged On	Steve Mohrman	SM
02/25/2010 11:12	Watchdog Message	Acknowledg	Process Media Gateway on Dev-01 has exited!	RAC
02/25/2010 10:59	Watchdog Message	Acknowledg	Process Media Gateway on Dev-01 has exited!	RAC
02/25/2010 10:57	Watchdog Message	Acknowledg	Process Media Gateway on Dev-01 has exited!	RAC
02/25/2010 10:42	PBX Assistant	Accept Call	Accept call for '(719) 439-9854 Rod Coles '	RAC
02/25/2010 10:36	Watchdog Message	Acknowledg	Process Media Gateway on Dev-01 has exited!	RAC
02/25/2010 10:29	PBX Assistant	Retrieve Call	Retrieve call for '(719) 439-9854 Rod Coles '	RAC
02/25/2010 10:28	PBX Assistant	Hold Call	Hold call for '(719) 439-9854 Rod Coles '	RAC
02/25/2010 10:28	PBX Assistant	Accept Call	Accept call for '(719) 439-9854 Rod Coles '	RAC
02/25/2010 10:28	PBX Assistant	Missed Call	Missed call for call session '100000726'	RAC
02/25/2010 10:23	PBX Assistant	Accept Call	Accept call for '(719) 439-9854 Rod Coles '	RAC
02/25/2010 10:22	PBX Assistant	Missed Call	Missed call for call session '100000724'	RAC
02/25/2010 10:21	Watchdog Message	Acknowledg	Process Media Gateway on Dev-01 has exited!	RAC
02/25/2010 10:13	PBX Assistant	Missed Call	Missed call for call session '100000723'	RAC
02/25/2010 10:10	Watchdog Message	Acknowledg	Process Media Gateway on Dev-01 has exited!	RAC
02/25/2010 09:47	PBX Assistant	Accept Call	Accept call for '(719) 439-9854 Rod Coles '	RAC
02/25/2010 09:45	PBX Assistant	Accept Call	Accept call for '(719) 439-9854 Rod Coles '	RAC
02/25/2010 09:45	PBX Assistant	Missed Call	Missed call for call session '100000719'	RAC
02/25/2010 09:45	PBX Assistant	Accept Call	Accept call for '(719) 439-9854 Rod Coles '	RAC
02/25/2010 09:45	Watchdog Message	Acknowledg	Process Media Gateway on Dev-01 has exited!	RAC
02/25/2010 09:36	PBX Assistant	Accept Call	Accept call for '(719) 439-9854 Rod Coles '	RAC
02/25/2010 09:36	Watchdog Message	Acknowledg	Process Media Gateway on Dev-01 has exited!	RAC
02/25/2010 09:12	PBX Assistant	Hold Call	Hold call for '(719) 439-9854 Rod Coles '	RAC
02/25/2010 09:11	PBX Assistant	Accept Call	Accept call for '(719) 439-9854 Rod Coles '	RAC
02/25/2010 09:04	PBX Assistant	Accept Call	Accept call for '(719) 439-9854 Rod Coles '	RAC
02/25/2010 09:04	PBX Assistant	Agent	Agent availability changed to 'Ready'	RAC
02/25/2010 09:03	Log On/Log Off	Logged On	Rod Coles	RAC

System Log

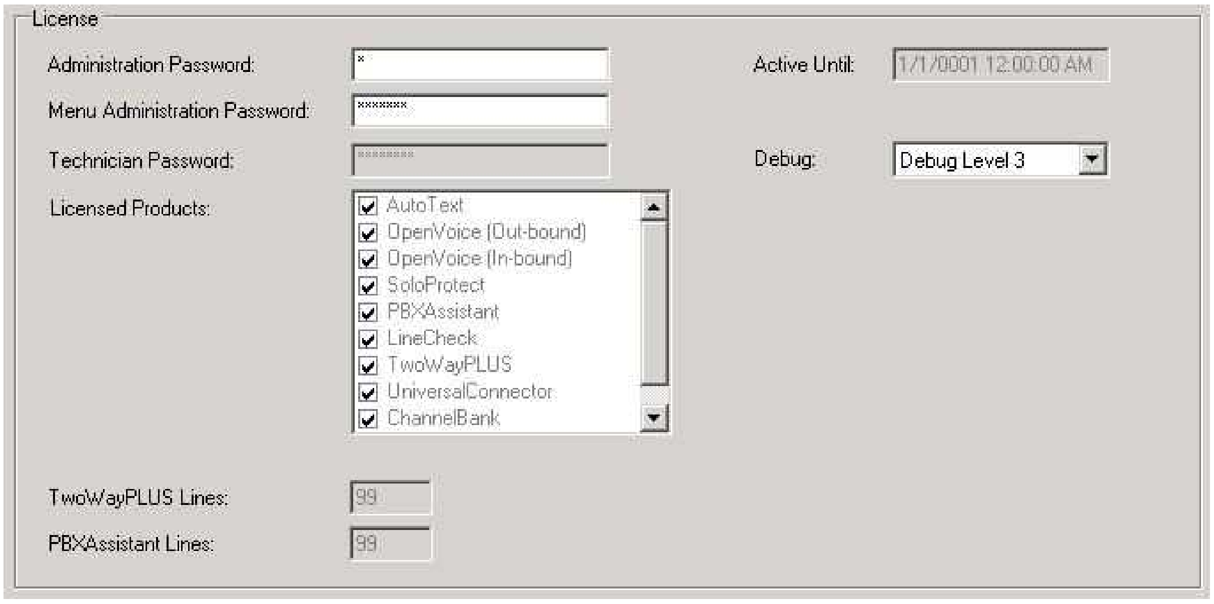
Options Menu

The MediaGateway 2 Options menu includes licensing and configuration options for the MediaGateway 2.



License

Use the MediaGateway 2 "License" form to enter passwords and verify the proper licensing of different MediaGateway 2 components. Licensing is handled during the initial MediaGateway 2 setup by a Bold technician using the Bold Licensing Tool. Therefore, the "License" form is mostly read-only. The Bold technician will also select a Debug mode to notify you regarding MediaGateway 2 issues and to help diagnose errors. Although it is possible to modify the "Debug:" field selection, we do not recommend such a modification.


License

Administration Password:	*
Menu Administration Password:	*******
Technician Password:	********
Licensed Products:	✓ AutoText ✓ OpenVoice (Out-bound) ✓ OpenVoice (In-bound) ✓ SoloProtect ✓ PBXAssistant ✓ LineCheck ✓ TwoWayPLUS ✓ UniversalConnector ✓ ChannelBank
TwoWayPLUS Lines:	99
PBXAssistant Lines:	99
Active Until:	1/1/0001 12:00:00 AM
Debug:	Debug Level 3

Options, License form

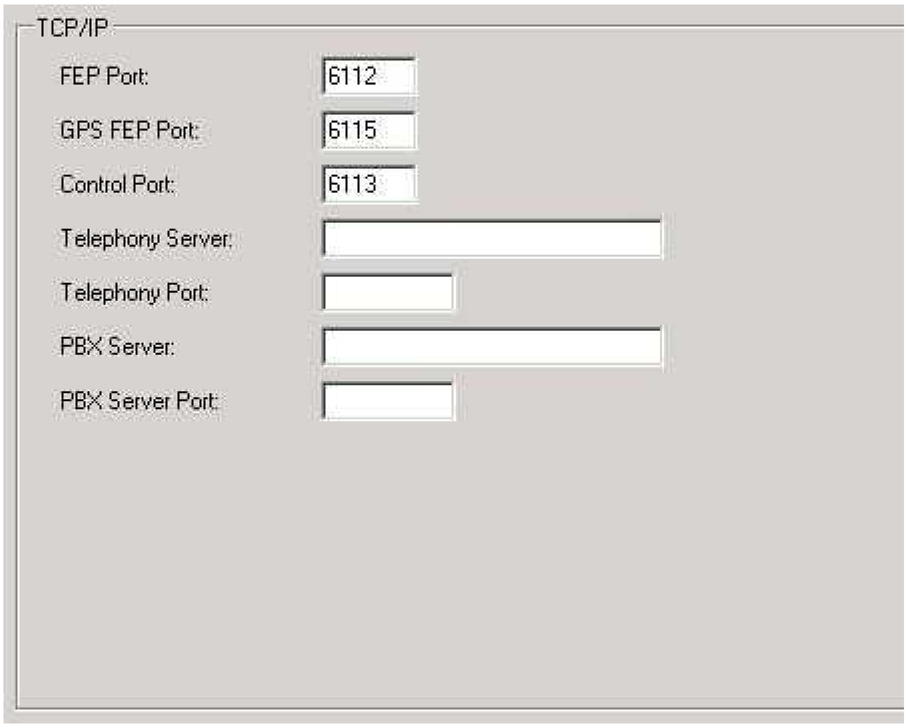
You can modify passwords for the "Administration Password:" and "Menu Administration Password:" fields.
Note: Two-WayPLUS voice service is licensed in packs of four lines each to provide a low entry price-point for the product.

TCP/IP

Use the TCP/IP options form to set up an FEP connection for the MediaGateway 2, and to define which ports connect with which functions.

- FEP Port - configure this port with a BOLDXML Driver that has an outbound TCP/IP connection to the MediaGateway server. The default port is 6112. This is the port that the MediaGateway 2 uses to listen for the XML Driver connection. It is an inbound TCP/IP connection.
- GPS FEP Port - any received signal that contains GPS coordinates passes through the GPS FEP. The default port is 6115.
- Control Port - this is the port to which the MediaGateway 2 talks to when it has been configured to run a server. The user interface makes an outbound connection to the Bold Communications Collection Service. The default port is 6113. This should not be modified.
- Telephony Server - this field contains the name or IP address of the server that is running the Bold Telephony Service. It is an outbound TCP/IP connection. This server

is used in conjunction with Dialogic cards.
- Telephony Port - the default port is 6110. This should not be modified.
- PBX Server - the PBX Server is used with a phone switch integration (PBX integration).


TCP/IP

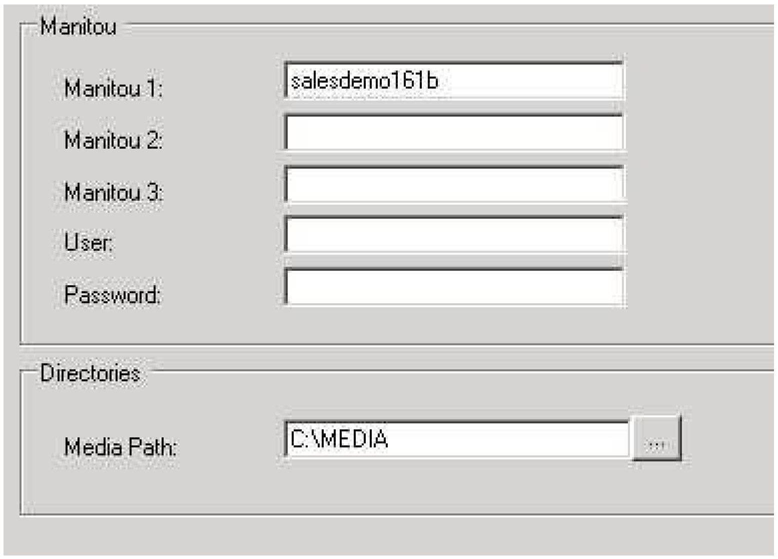
FEP Port:	6112
GPS FEP Port:	6115
Control Port:	6113
Telephony Server:	
Telephony Port:	
PBX Server:	
PBX Server Port:	

Options, TCP/IP form

Note: leaving a field on the TCP/IP form blank disables that particular function in the MediaGateway 2.

Configuration

Use the Configuration options form to define the machine on which Manitou is currently running.


Manitou

Manitou 1:	salesdemo161b
Manitou 2:	
Manitou 3:	
User:	
Password:	

Directories

Media Path:	C:\MEDIA

Options, Configuration form

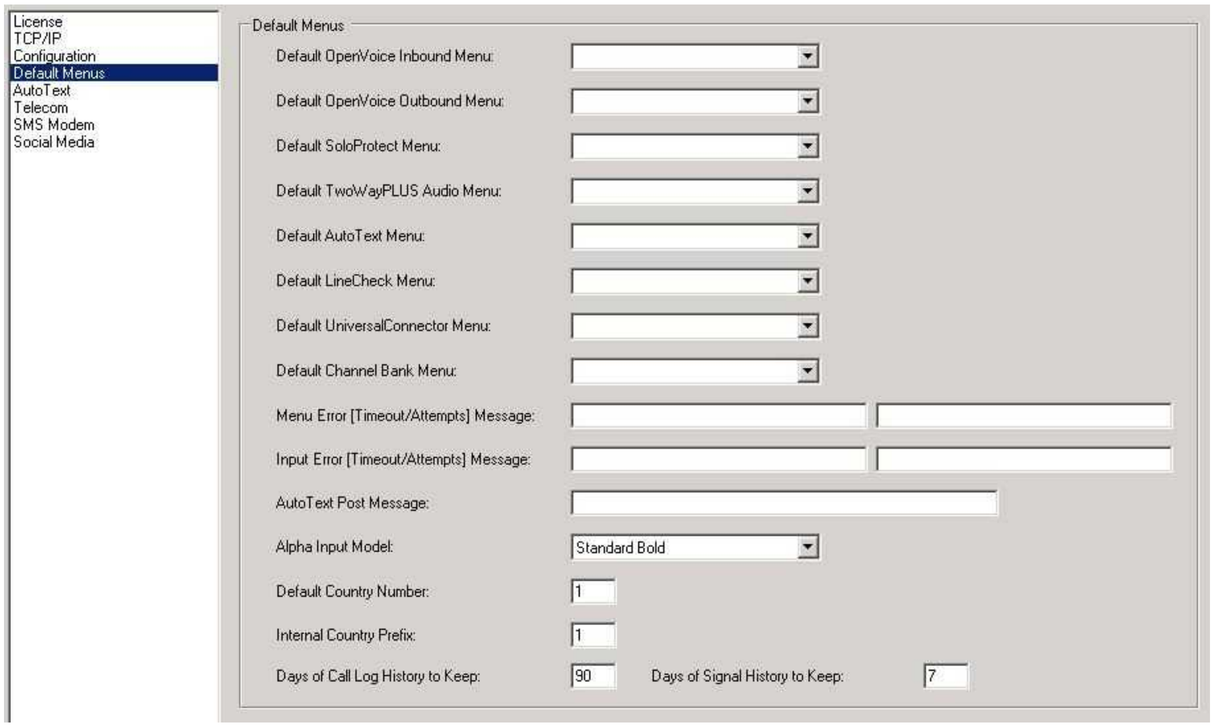
You can define up to three Manitou servers to which the MediaGateway 2 will attempt to connect. The MediaGateway 2 remembers the last Manitou server to which it connected, and will attempt to reconnect to that server first. If it fails to connect, Manitou then attempts to connect to the next server on the list until it finds an active Manitou system.

The "User:" and "Password:" fields contain passwords with which you can log into Manitou as a full user. This is not required in most cases, because the typical user connects using a MediaGateway 2 login. However, there may be requirements for which you want to connect to Manitou (for e.g., when you experience a bad connection and need to access a caller ID). Logging into Manitou through the MediaGateway 2 transmits as a signal to the account. Therefore, connecting in this manner should be limited.

The Media Path is the directory where voice clips are stored. The MediaGateway 2 creates a voice clip whenever it reads a menu. To maintain efficiency and to prevent the system from having to generate a text-to-speech menu each time a menu is read, the system plays a common .wav file stored in the Media Path location. We recommend you create a shared directory, because it may be necessary to switch between primary and secondary systems. Also, the system creates a binary object from these voice clips. The MediaGateway 2 then sends the voice clips into Manitou, where they are logged.

Default Menus

Default menus make it possible for a function to still perform and operate as expected when no other menu has been designated. Although, it is not necessary to select default menus, it is highly recommended.



Default Menus

Default OpenVoice Inbound Menu:

Default OpenVoice Outbound Menu:

Default SoloProtect Menu:

Default TwoWayPLUS Audio Menu:

Default AutoText Menu:

Default LineCheck Menu:

Default UniversalConnector Menu:

Default Channel Bank Menu:

Menu Error [Timeout/Attempts] Message:

Input Error [Timeout/Attempts] Message:

AutoText Post Message:

Alpha Input Model: Standard Bold

Default Country Number: 1

Internal Country Prefix: 1

Days of Call Log History to Keep: 90 Days of Signal History to Keep: 7

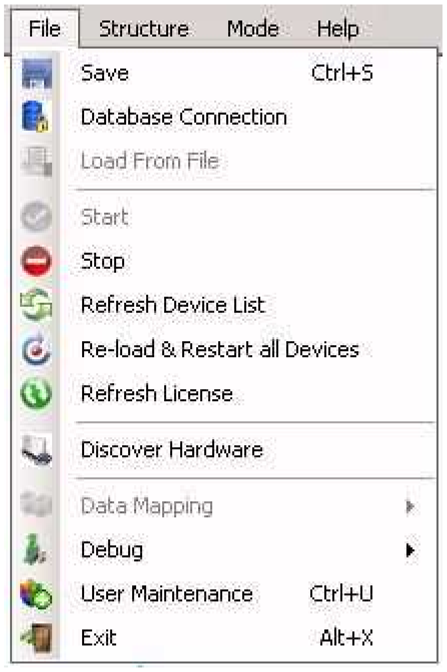
Setting Default Menus

Perform the following steps to set default menus in the MediaGateway 2:

1. Click "Options" on the MediaGateway 2 Navigation Tree.
2. Click "Default Menus".
3. Click the arrows for the dropdown menus to the right of the fields, and then select the default menus for each of the MediaGateway 2 components you are setting up.

Note: if the menu you want is not listed in the dropdown list, you may want to see if it is available for import. Refer to the "Importing and Exporting Menus" document for more information on how to import a menu.

4. Navigate to the File menu, and then click "**Save**".
5. In the File menu, click "Re-load & Restart all Devices" as shown in the following screenshot:



Default Menus: Additional Fields

The following are Default Menu items not associated with specific MediaGateway 2 components:

- Menu Error [Timeout/Attempts] Message - message announced when a timeout or too many attempts generate a menu error
- Input Error [Timeout/Attempts] Message - message announced when a timeout or too many attempts generate an input error
- AutoText Post Message - common text appended to an SMS message intended to guide or help a user respond
- Alpha Input Model - select which code model you want to use for texting
- Default Country Number - select a country code to use in cases where international dialing rules apply
- Internal Country Prefix - provide an internal country prefix to ensure that the incoming caller ID matches the outgoing caller ID
- Days of Call Log History to Keep - specify the number of days you want to keep a record of Call Log History
- Days of Signal History to Keep - specify the number of days you want to keep historical signal data

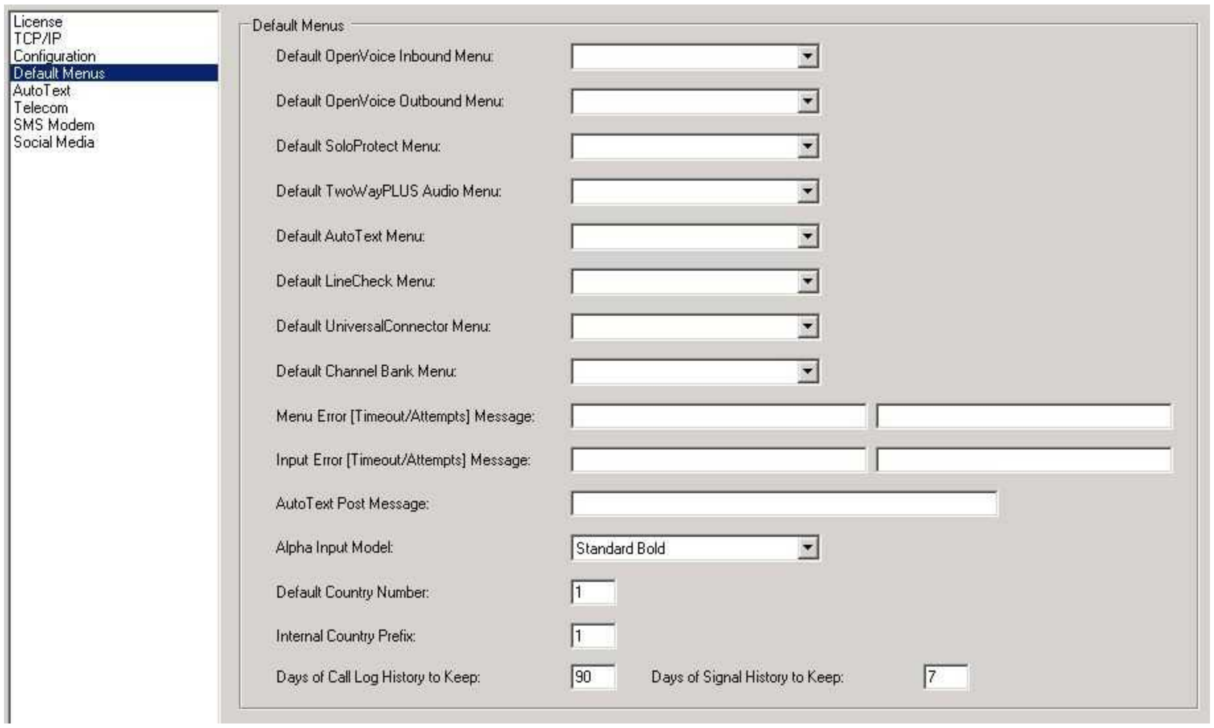
Alpha Codes

Alpha Codes are used to identify which Alpha Input Model you want to use for SMS texting.

Perform the following steps to define an Alpha Input Model:

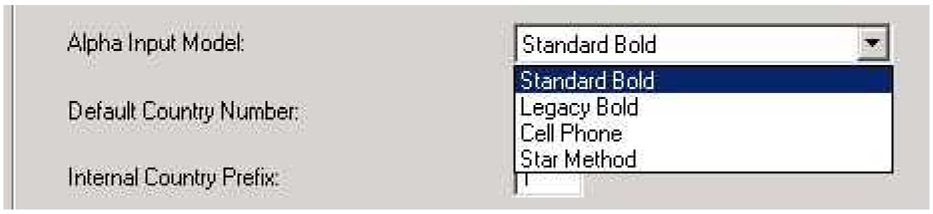
1. In the MediaGateway 2, click "Options" in the Navigation Tree.
2. Click "Default Menus".

Result: the Default Menus form displays as shown in the following screenshot:



Default OpenVoice Inbound Menu:	
Default OpenVoice Outbound Menu:	
Default SoloProtect Menu:	
Default TwoWayPLUS Audio Menu:	
Default AutoText Menu:	
Default LineCheck Menu:	
Default UniversalConnector Menu:	
Default Channel Bank Menu:	
Menu Error [Timeout/Attempts] Message:	
Input Error [Timeout/Attempts] Message:	
AutoText Post Message:	
Alpha Input Model:	Standard Bold
Default Country Number:	1
Internal Country Prefix:	1
Days of Call Log History to Keep:	90
Days of Signal History to Keep:	7

3. Select an option from the "Alpha Input Model:" dropdown menu as shown in the following screenshot:

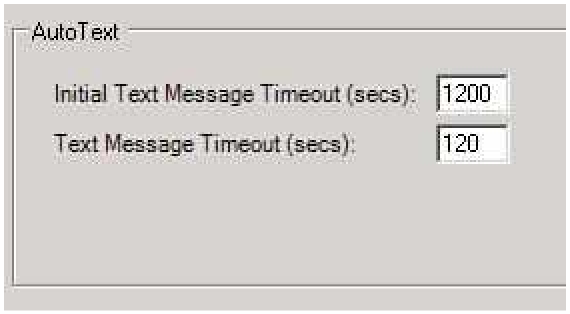

Alpha Input Model: Standard Bold

Option Description	Examples	
Standard Bold - Uses an asterisk, "*", then using the telephone keypad representation the user chooses the first digit based on the location of the letter (i.e. 2 references A, B or C, 3 references D, E or F, etc.) and the second digit as the location of the letter in reference to the first.	*21 = A *22 = B	*32 = E *33 = F
Legacy Bold - Uses an asterisk, "*", followed by a two digit number representing the letter. The numbers range from 01 to 26.	*01 = A *02 = B	*03 = C *26 = Z
Cell Phone - Always uses an asterisk, "*", to start and then number key on the telephone keypad to represent the digit or character needed:	*2 = 2 *22= A *3= 3 *333= E	*222= B *2222= C *33= D
Star Method - Uses a series of asterisks, "*", to represent the location of the letter in reference to a single keypad digit and then the location of the letter based on the telephone keypad representation of characters.	*2= A **2= B ***2= C	*3= D **3= E

4. Click "**Save**" from the File menu.

AutoText

The AutoText form has only two fields - the Initial Text Message Timeout (in seconds) and the Text Message Timeout (in seconds).



Initial Text Message Timeout (secs): 1200
Text Message Timeout (secs): 120

Options, AutoText form

Initial Text Message Timeout: the time that must pass between the initial text message and a response. The default value is 1200.

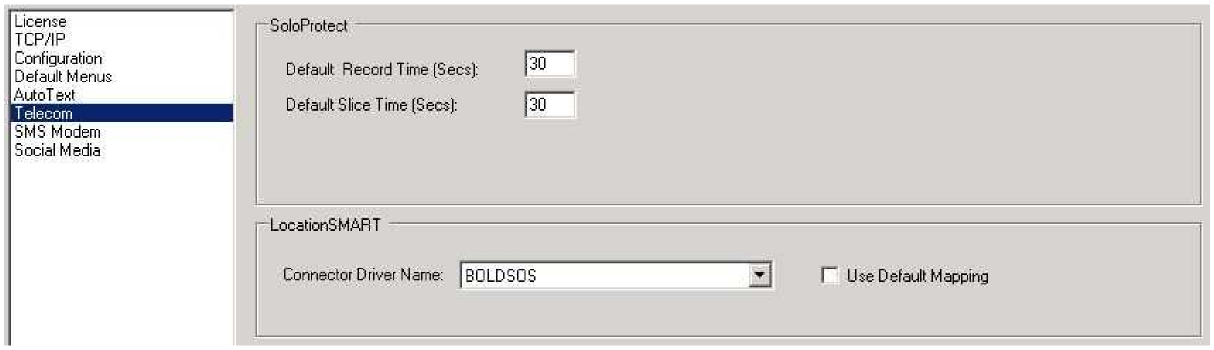
Text Message Timeout: the time by which a response must be received in order to avoid a timeout. The default value is 120.

Telecom

SoloProtect

SoloProtect uses caller ID information to create an alarm with voice recording. SoloProtect works with streaming audio and recorded audio slices to help apprise Operators of emergency situations without the need for a 2-way conversation.

The SoloProtect form in the MediaGateway 2 Options menu allows you to set the default times (in seconds) for recording and slicing audio.



Default Record Time (Secs): 30
Default Slice Time (Secs): 30
Connector Driver Name: BOLDSOS

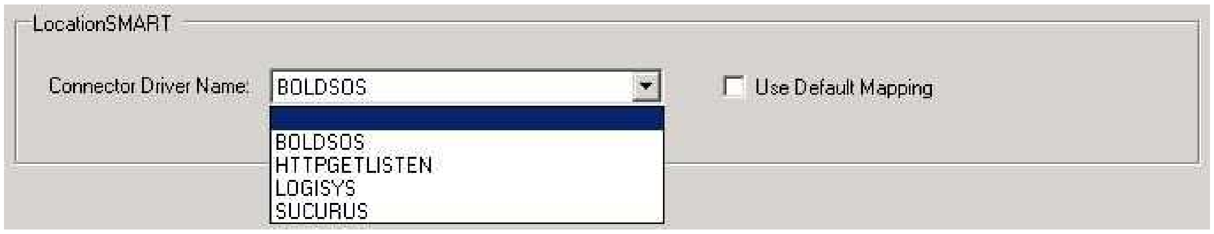
Default Record Time: the total number of seconds you want to allow for voice recording. The default value is 30.

Default Slice Time: the total number of seconds for each slice of the total record time. The default value is 30.
(For e.g., if the record time is set to 30 seconds and the slice time is set to 10 seconds, then you will derive 3 10-second audio slices. The default values yield only a single audio slice.

LocationSMART

LocationSMART is a service that provides geographic coordinates of a user's location.

Select a Connector Driver Name from the dropdown menu. As shown in the following screenshot, you have the following options:

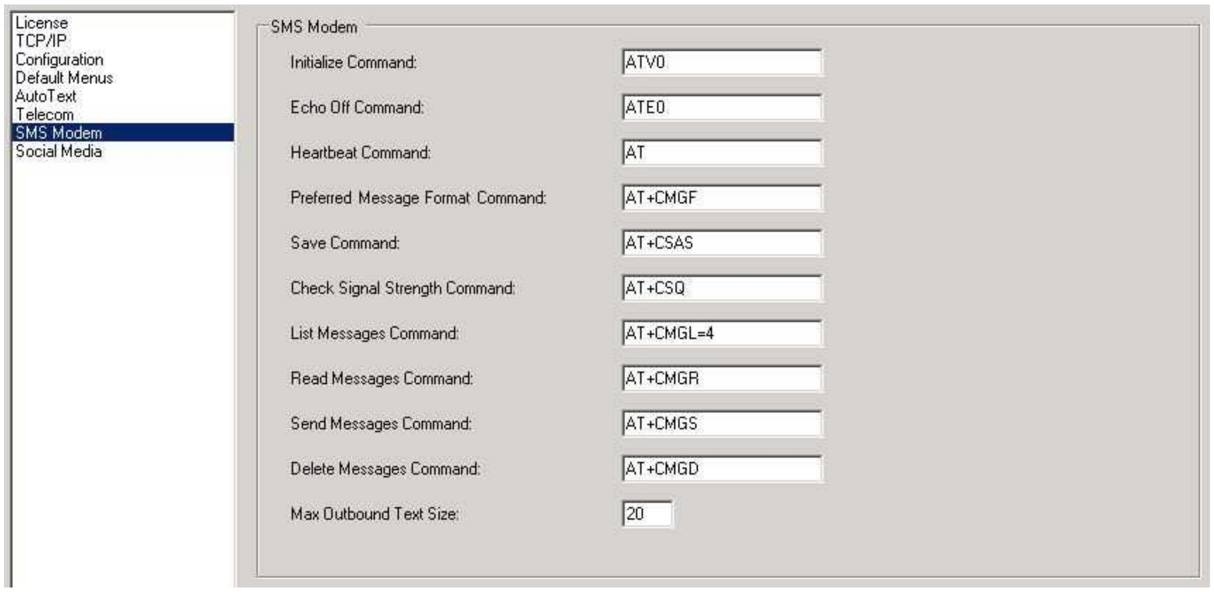

LocationSMART

Connector Driver Name: BOLDSOS

☐ Use Default Mapping

SMS Modem

The "SMS Modem" form contains information necessary for connection to SMS modems. You can modify the information in these fields. However, we strongly recommend leaving them as shown in the following screenshot:



SMS Modem

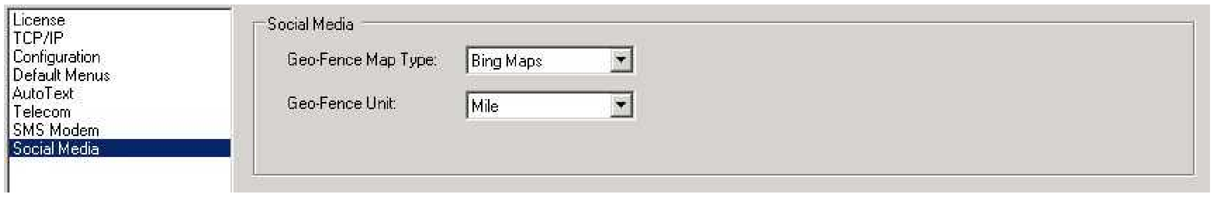
Initialize Command:	ATV0
Echo Off Command:	ATE0
Heartbeat Command:	AT
Preferred Message Format Command:	AT+CMGF
Save Command:	AT+CSAS
Check Signal Strength Command:	AT+CSQ
List Messages Command:	AT+CMGL=4
Read Messages Command:	AT+CMGR
Send Messages Command:	AT+CMGS
Delete Messages Command:	AT+CMGD
Max Outbound Text Size:	20

While Manitou and the MediaGateway 2 still support SMS modems, this section is included for Legacy use only.

Social Media

The MediaGateway 2 offers the ability to monitor Social Media outlets for negative, malicious, and brand-related content.

To access the MediaGateway 2 Social Media form, click "Options", and then select "Social Media" as shown in the following screenshot:

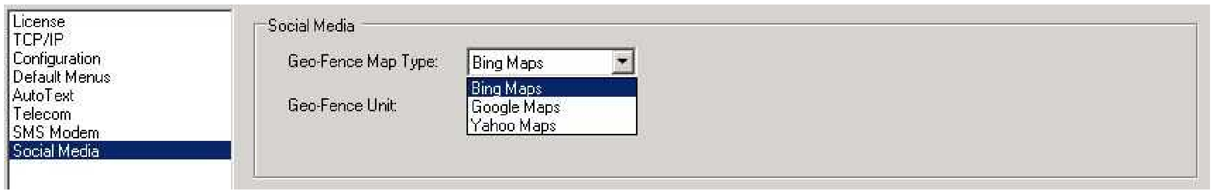


Social Media

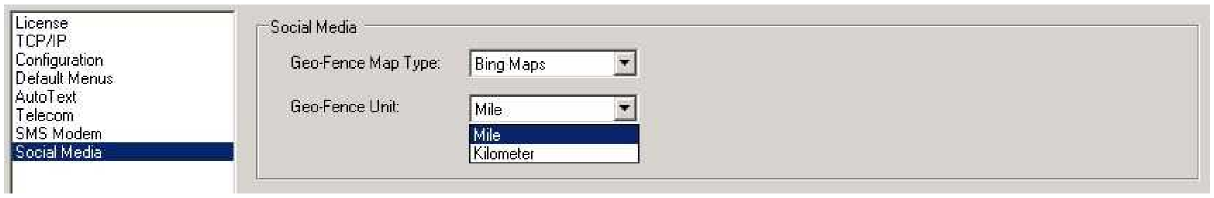
Geo-Fence Map Type: Bing Maps

Geo-Fence Unit: Mile

Geo-Fences are user-defined boundaries that denote a geographic search area. They are used with Social Media Monitoring, and detect postings to Facebook and Twitter within the defined area. Select a Geo-Fence Map Type from the dropdown menu. You have the options displayed in the following screenshot:



Geo-Fences can be set to radiate up to 100 miles from a central point. The default unit for Geo-Fences is miles. If you want Geo-Fences to to be measured in kilometers instead, select that option from the dropdown menu.



Reverse Commands

A new Reverse Command type called MediaGateway is now available in Manitou. The following six Reverse Commands are available under the Central Station's Reverse Commands menu:

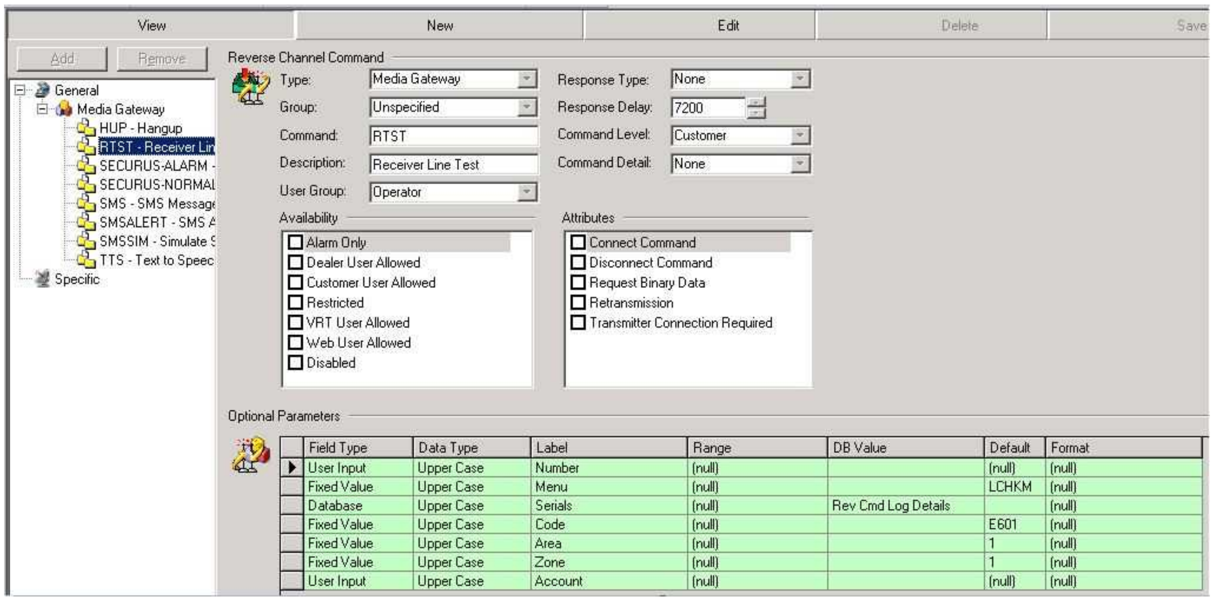
- HUP, for Hang-up

- RTST, for LineCheck’s receiver line testing
- SMS, for AutoText’s SMS-sending capability
- SMSSIM, to simulate sending an SMS
- TTS, for OpenVoice outbound Text-to-Speech

Perform the following steps to access the Reverse Channel Command form:

1. Open the Manitou Operator Workstation.
2. Open the Customer for whom you want to define Reverse Commands.
3. Select "Reverse Commands" from the Jump To: menu.

Result: the "Reverse Channel Command" form displays as shown in the following screenshot:



Reverse Channel Command

Type:	Media Gateway	Response Type:	None
Group:	Unspecified	Response Delay:	7200
Command:	RTST	Command Level:	Customer
Description:	Receiver Line Test	Command Detail:	None
User Group:	Operator		

Availability

- ☐ Alarm Only
- ☐ Dealer User Allowed
- ☐ Customer User Allowed
- ☐ Restricted
- ☐ VRT User Allowed
- ☐ Web User Allowed
- ☐ Disabled

Attributes

- ☐ Connect Command
- ☐ Disconnect Command
- ☐ Request Binary Data
- ☐ Retransmission
- ☐ Transmitter Connection Required

Optional Parameters

Field Type	Data Type	Label	Range	DB Value	Default	Format
User Input	Upper Case	Number	(null)		(null)	(null)
Fixed Value	Upper Case	Menu	(null)		LCHKM	(null)
Database	Upper Case	Serials	(null)	Rev Cmd Log Details		(null)
Fixed Value	Upper Case	Code	(null)		E601	(null)
Fixed Value	Upper Case	Area	(null)		1	(null)
Fixed Value	Upper Case	Zone	(null)		1	(null)
User Input	Upper Case	Account	(null)		(null)	(null)

Reverse Channel Routes and Service Provider Devices

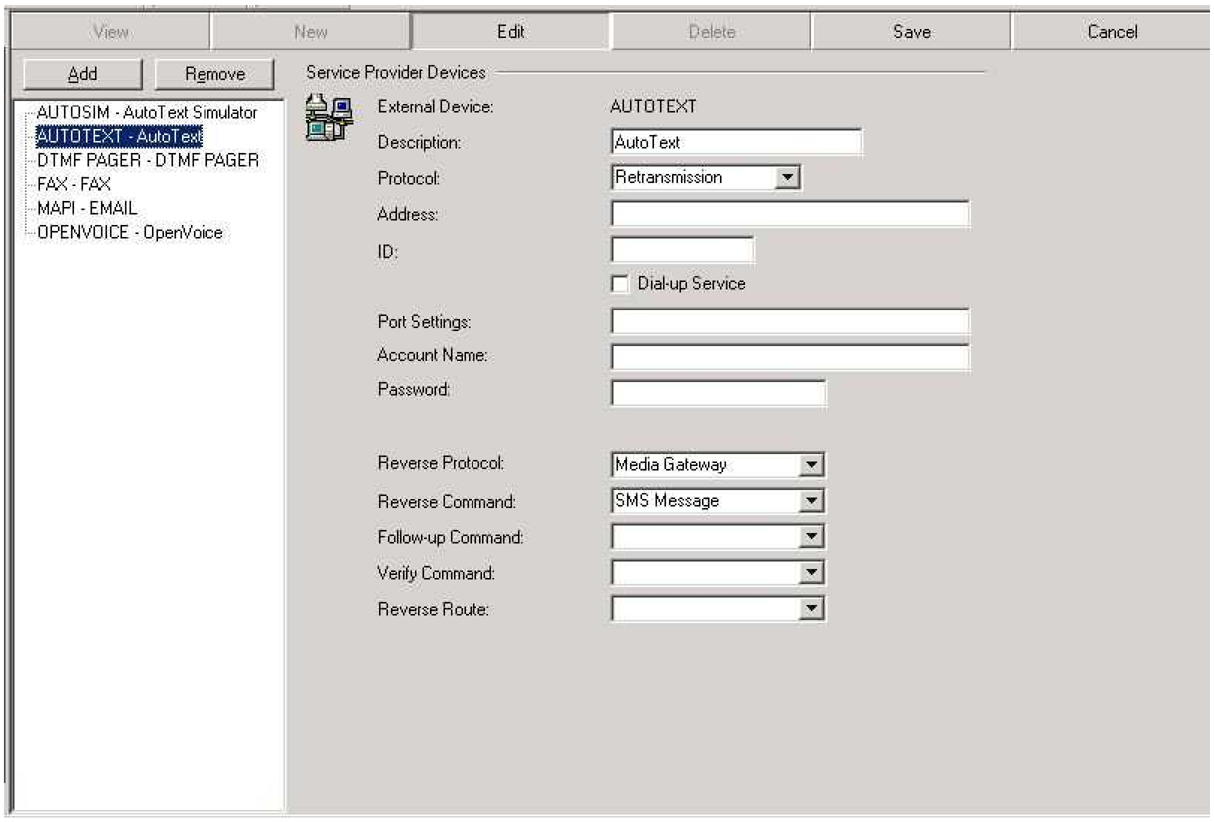
Additionally, a new Reverse Channel Route (MediaGateway) is also available, along with three Service Provider Devices:

- OPENVOICE for OpenVoice
- AUTOTEXT for AutoText
- AUTOSIM for AutoText Simulator

Perform the following steps to access the Service Provider Devices form:

1. Open the Manitou Supervisor Workstation.
2. Navigate to the Maintenance menu, and then click "Setup", and "Service Provider

Devices".
Result: the "Service Provider Devices" form displays as shown in the following screenshot:



AUTOSIM - AutoText Simulator
AUTOTEXT - AutoText
DTMF PAGER - DTMF PAGER
FAX - FAX
MAPI - EMAIL
OPENVOICE - OpenVoice

Service Provider Devices

External Device:	AUTOTEXT
Description:	AutoText
Protocol:	Retransmission
Address:	
ID:	
	☐ Dial-up Service
Port Settings:	
Account Name:	
Password:	
Reverse Protocol:	Media Gateway
Reverse Command:	SMS Message
Follow-up Command:	
Verify Command:	
Reverse Route:	

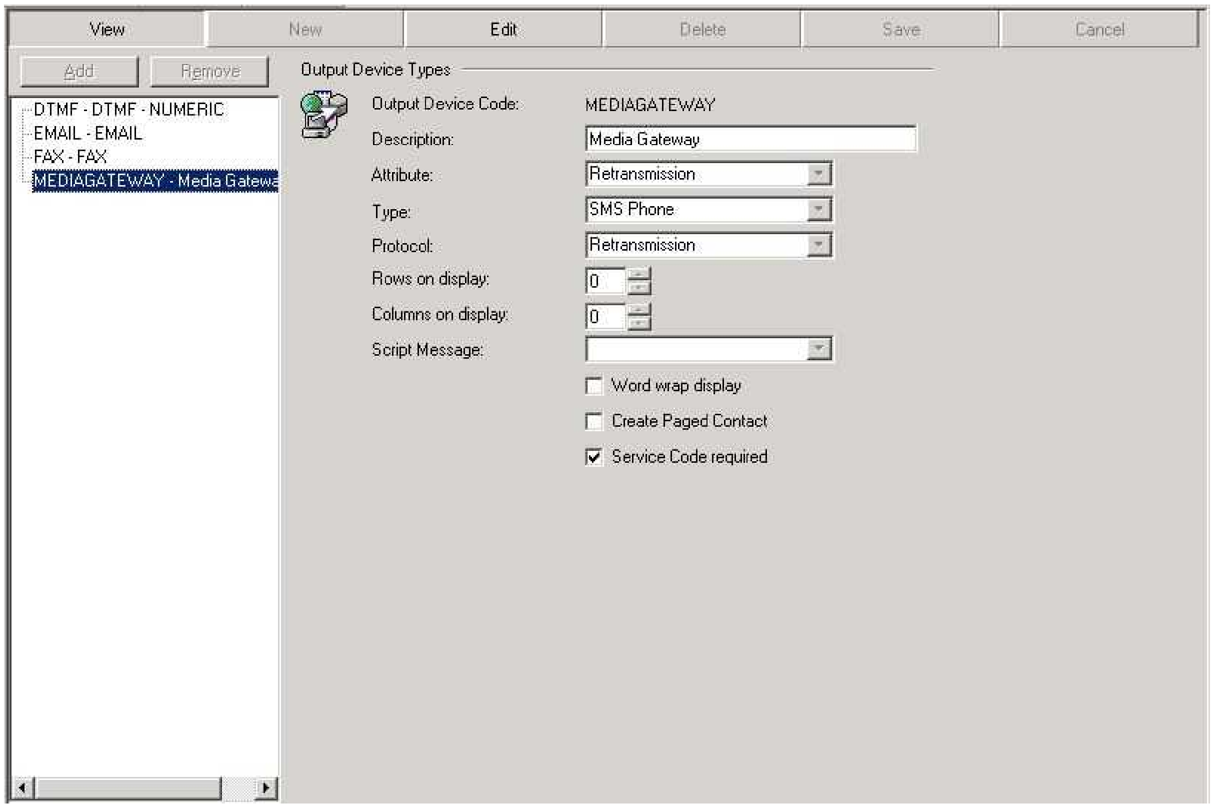
When creating a new Service Provider Device for the MediaGateway 2, the Protocol must be set to "Retransmission" and the Reverse Protocol must be "Media Gateway".

Output Device Types

Similarly, there is an Output Device Type called MediaGateway.

Perform the following steps to access the Output Device Types form:

1. Open the Manitou Supervisor Workstation.
2. Navigate to the Maintenance menu, and then click "Setup", and "Output Device Types".
Result: the "Output Device Types" form displays as shown in the following screenshot:



DTMF - DTMF - NUMERIC
EMAIL - EMAIL
FAX - FAX
MEDIAGATEWAY - Media Gatewa

Output Device Types

Output Device Code:	MEDIAGATEWAY
Description:	Media Gateway
Attribute:	Retransmission
Type:	SMS Phone
Protocol:	Retransmission
Rows on display:	0
Columns on display:	0
Script Message:	

☐ Word wrap display
☐ Create Paged Contact
☑ Service Code required

When creating a new Output Device Type for the MediaGateway 2, the Attribute and Protocol must be "Retransmission," and the Type must be "SMS Phone". Also, select the "Service Code Required" checkbox.

Contact Point Types

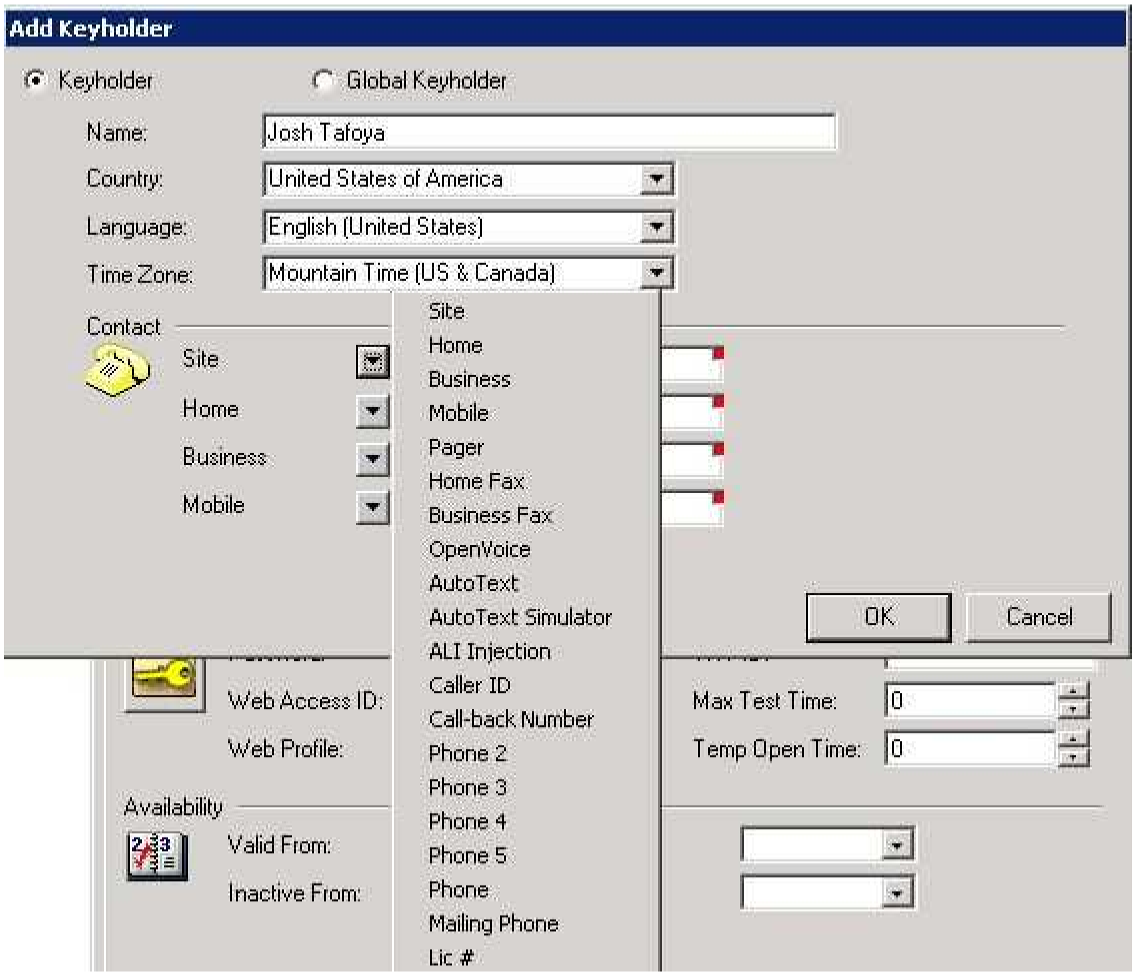
The MediaGateway 2 includes the following three Contact Point Types:

- OpenVoice
- AutoText
- AutoText Simulator

Perform the following steps to access the "Contact Point Types" form:

1. Open the Manitou Supervisor Workstation.
2. Navigate to the Maintenance menu, and then click "Setup", and "Contact Point Types".

Result: the "Contact Point Types" form displays as shown in the following screenshot:



Name: Josh Tafoya
Country: United States of America
Language: English (United States)
Time Zone: Mountain Time (US & Canada)

Contact
Site
Home
Business
Mobile

Web Access ID:
Web Profile:
Max Test Time: 0
Temp Open Time: 0

Availability
Valid From:
Inactive From:

When creating new Contact Point Types, the Attributes must be set to "Retransmission."

A default script is optional for the MediaGateway 2 Contact Points. Creating a script for the MediaGateway 2 is similar to creating normal Manitou scripts. In order to specify an individual MediaGateway 2 menu in the output script, however, the following syntax is required: "$MC<media gateway menu code>" (where $MC specifies a MediaGateway menu).

You can use the MediaGateway 2 in Action Patterns to send SMS messages and make outbound calls. If you want the Action Pattern to be automated, you must specify a script for the Contact Points at the Contact Point level. Please see the Manitou Operator Workstation manual for instructions on how to set up Action Patterns.

MediaGateway 2 Components

The MediaGateway 2 is a compilation of independent products working together to provide unique solutions to Central Station business problems. These products are grouped into two modules. These can be individually licensed as either the Bold Efficiency Collection or the Personal Safety Collection, or combined.

Bold Efficiency Collection Components

OpenVoice - allows users to build a system of customized voice responses for unique business needs using text-to-speech technology. The system is inbound and outbound voice response capable, and can be tailored to caller ID or DNIS. Outbound OpenVoice may be used in conjunction with the AutoClient and text-to-speech for automated notification of alarms and Customer interactive responses.

AutoText - uses true SMS messaging technology for notification of alarms and signals, as well as for two-way Customer interaction. The module applies the same intuitive menus as OpenVoice, so users can create a single menu that can be used for either voice or text responses. Likewise, users can create a customizable menu specifically for text or email responses. When combined with AutoClient, AutoText has the ability to handle alarms via text message without Operator intervention. For example, a low-battery signal arrives while the Customer is out of town. The Customer receives notification of the signal via text or email message, and can then respond with authentication to cancel the alarm without any Operator involvement.

LineCheck - tests Receiver lines and reports any failures. Additionally, you can configure a separate Receiver to report LineCheck failures. Users also have the option of running Reports to view all LineCheck activity.

Virtual Operator - enables Manitou to automatically respond to an alarm event by handling the assigned signals and sending the necessary responses without Operator intervention. The system can even log, suspend, or close out alarms.

AutoClient - enables Manitou to automatically respond to an alarm event by handling signals and alarms via e-mail, fax, text messaging, Reverse Commands, and Text-To-Speech messages without Operator involvement.

PBX Assistant - the PBX Assistant works with OpenVoice to provide full call control with interactive voice response and routing using automatic call distribution. This allows the most appropriate Operator to be located by communication with the Manitou system whenever a call arrives. It also ensures that Operators do not receive calls unrelated to the alarms they are currently handling.

Channel Bank - allows you to connect your PRI/T1 directly into the MediaGateway 2 station ports. This allows Manitou to receive an alarm while automatically busying out the line, and to detect conditions such as no answer.

Social Media Monitoring - monitors Twitter and Facebook for information that could trigger an alarm. Through authorizing an account on Twitter, Social Media Monitoring can send tweets as well as monitor by search criteria within a specified area, using Geo-Fencing. While Facebook does not currently allow postings from the MediaGateway 2, the module can still monitor current postings using ranked search requests and Geo-Fencing specifications.

UniversalConnector - connects a variety of different delivery mechanisms in a one-stop location for easy setup and management. The UniversalConnector allows signals to be delivered through PRI/T1/Analog, the PBX Server, SMS, email, ODBC database, FTP, TCP, RSS, and simple files.

Personal Safety Collection Components

TwoWayPLUS - combines TwoWayVoice with InstantConnect components.

- **TwoWayVoice** - with TwoWayVoice, an incoming call is processed at the Receiver and sent to the Central Station Operator at the same time as the alarm presents in Manitou. The Manitou user interface controls two-way functions such as talk and listen capabilities so there is one consistent interface for Operators, regardless of the panel. Additionally, Manitou automatically manages the panel "keep alive." All two-way activity is recorded in Manitou, and if used in conjunction with SecurVoice, the two-way call can also be stored in Manitou.

- **InstantConnect** - enables any phone to become an alarm system. When a telephone call is received at the Central Station, InstantConnect uses caller ID and generates an alarm in Manitou. Then, when the Operator picks up the alarm, the voice call is routed to the Operator. An alarm can be created and routed to an Operator within two or three rings. Then, the call follows the alarm when it passes to another Operator. InstantConnect uses caller ID in place of the Transmitter ID, so no Receiver is required.

SoloProtect - is designed for lone worker safety and protection. Using caller ID, SoloProtect creates an alarm in Manitou with a recorded voice clip. This module integrates with any phone device that has caller ID.

AutoText

The AutoText feature provides the ability to automatically text notifications, use script messages, and to initiate and receive responses from a texting platform. In order to set up the AutoText feature, it is first necessary to create a script. This enables text-back capabilities when the MediaGateway 2 is engaged.

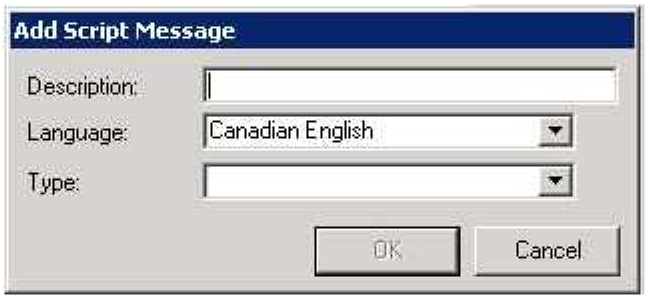
Creating the Script

Scripts are created and managed in the Manitou Supervisor Workstation.

Perform the following steps to add a script:

1. Open the Supervisor Workstation
2. Navigate to the Maintenance menu, and then select "Script Messages".
3. Click "**Edit**", and then click "**Add**".

Result: the "Add Script Message" displays as shown in the following screenshot:



Add Script Message

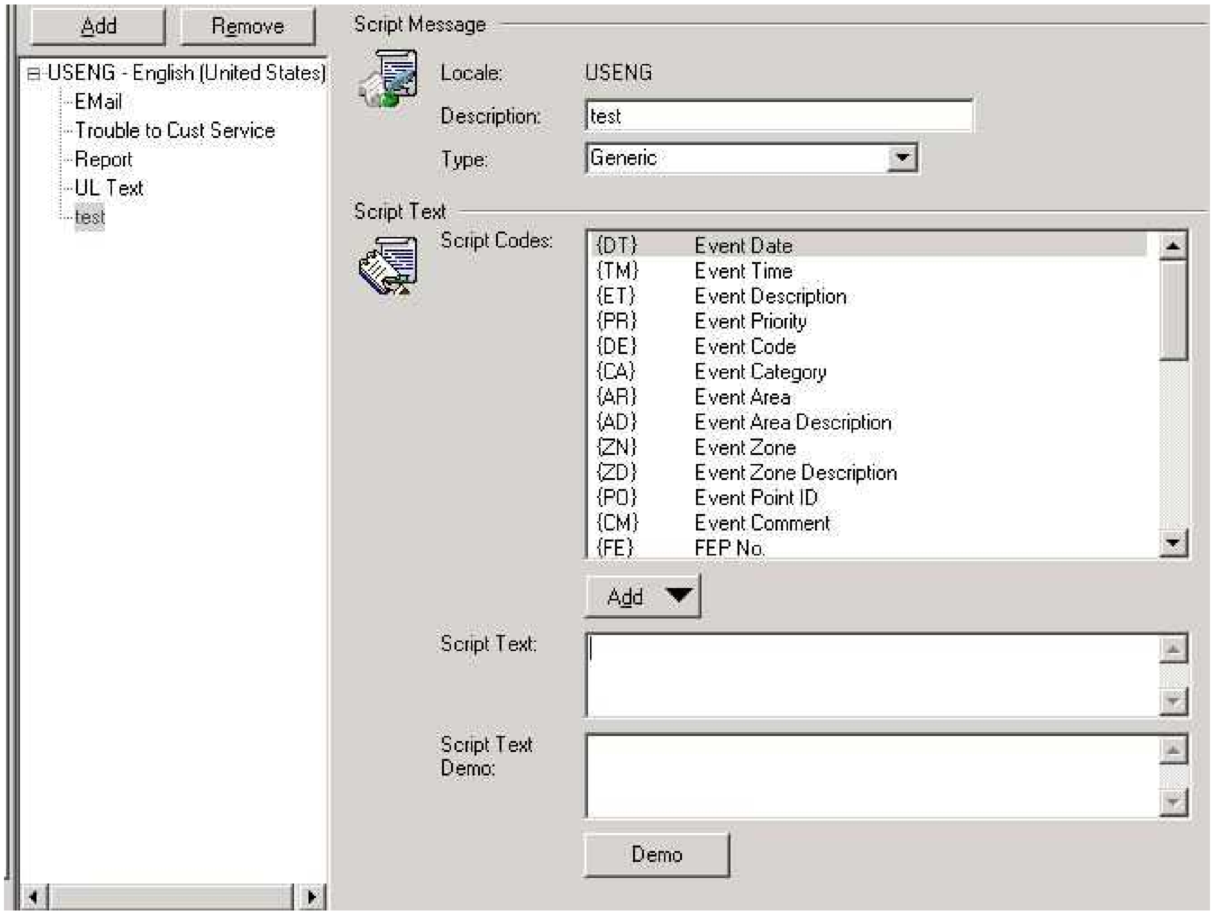
Description:

Language: Canadian English

Type:

4. Enter a description, and select a language from the "Language:" dropdown menu.
5. Select "Generic" from the "Type:" dropdown menu, and then click "**OK**".

Result: the "Add Script Message" window closes and the "Script Message" window displays as shown in the following screenshot:



Script Message

Locale: USENG

Description: test

Type: Generic

Script Text

Script Codes:

{DT}	Event Date
{TM}	Event Time
{ET}	Event Description
{PR}	Event Priority
{DE}	Event Code
{CA}	Event Category
{AR}	Event Area
{AD}	Event Area Description
{ZN}	Event Zone
{ZD}	Event Zone Description
{PO}	Event Point ID
{CM}	Event Comment
{FE}	FEP No.

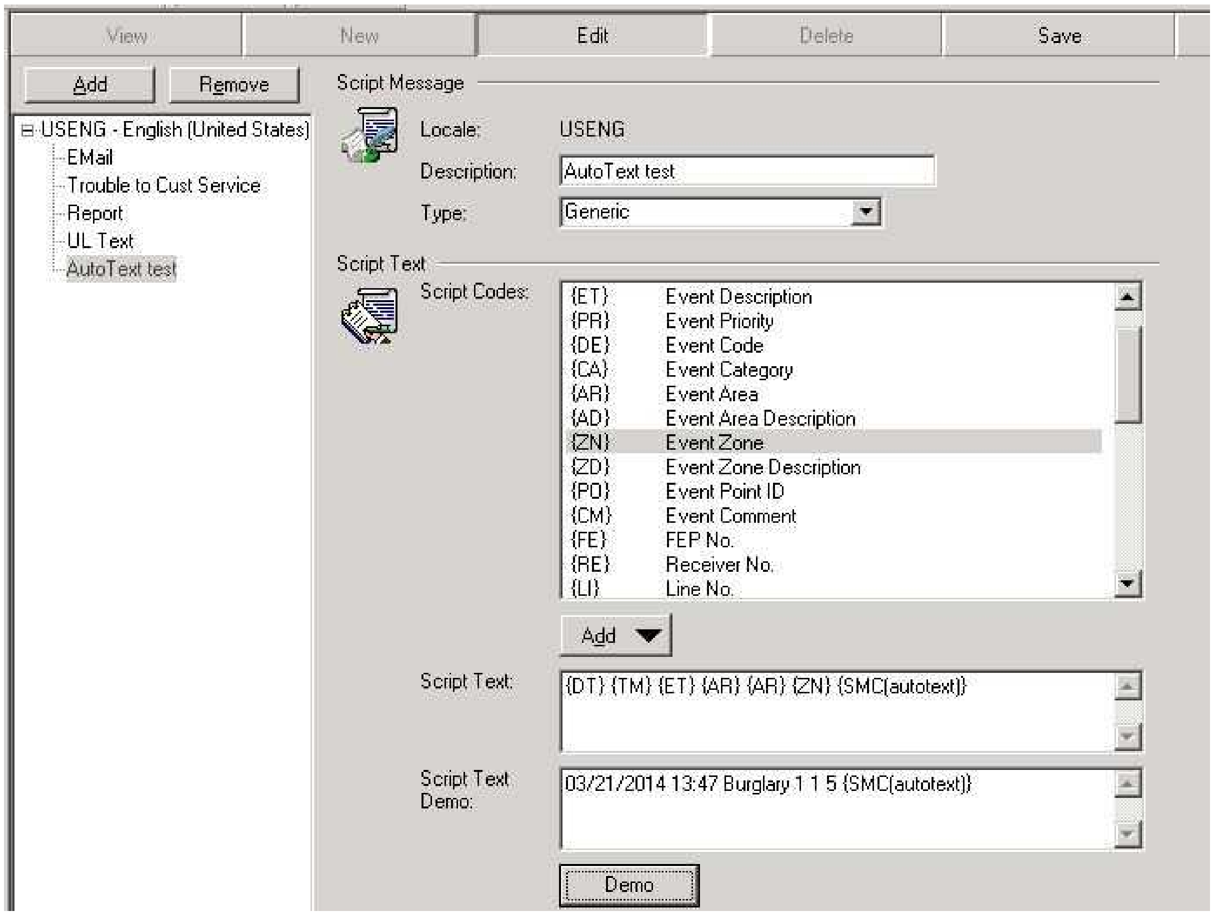
Script Text:

Script Text Demo:

6. Select the codes you want from the "Script Codes:" window, and then add them to the "Script Text:" window using the "**Add**" button.
7. Enter the text for your script message into the "Script Text:" window.
8. Enter the following code at the end of your script message text: "{SMC(menu name)}". Insert the actual MediaGateway 2 menu name into the code.

Note: this ensures that the MediaGateway 2 is engaged for the AutoText feature.

9. If you want, click "**Demo**" to show how the script message will display to your recipient.



USENG - English (United States)
- EMail
- Trouble to Cust Service
- Report
- UL Text
- AutoText test

Locale: USENG
Description: AutoText test
Type: Generic

Script Codes:

{ET}	Event Description
{PR}	Event Priority
{DE}	Event Code
{CA}	Event Category
{AR}	Event Area
{AD}	Event Area Description
{ZN}	Event Zone
{ZD}	Event Zone Description
{PO}	Event Point ID
{CM}	Event Comment
{FE}	FEP No.
{RE}	Receiver No.
{LI}	Line No.

Script Text: {DT} {TM} {ET} {AR} {AR} {ZN} {SMC(autotext)}

Script Text Demo: 03/21/2014 13:47 Burglary 1 1 5 {SMC(autotext)}

10. Click "**Save**".

Applying the Script

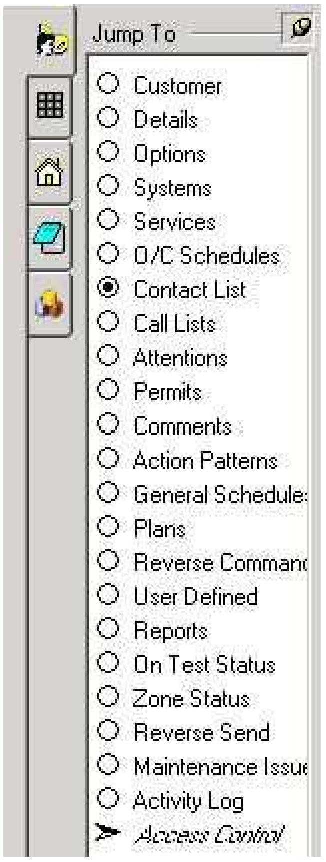
Once you have created a script in the Supervisor Workstation, you must designate the Contact Point phone number for which you want to use AutoText in the Operator Workstation.

Designating the Contact Point Phone Number

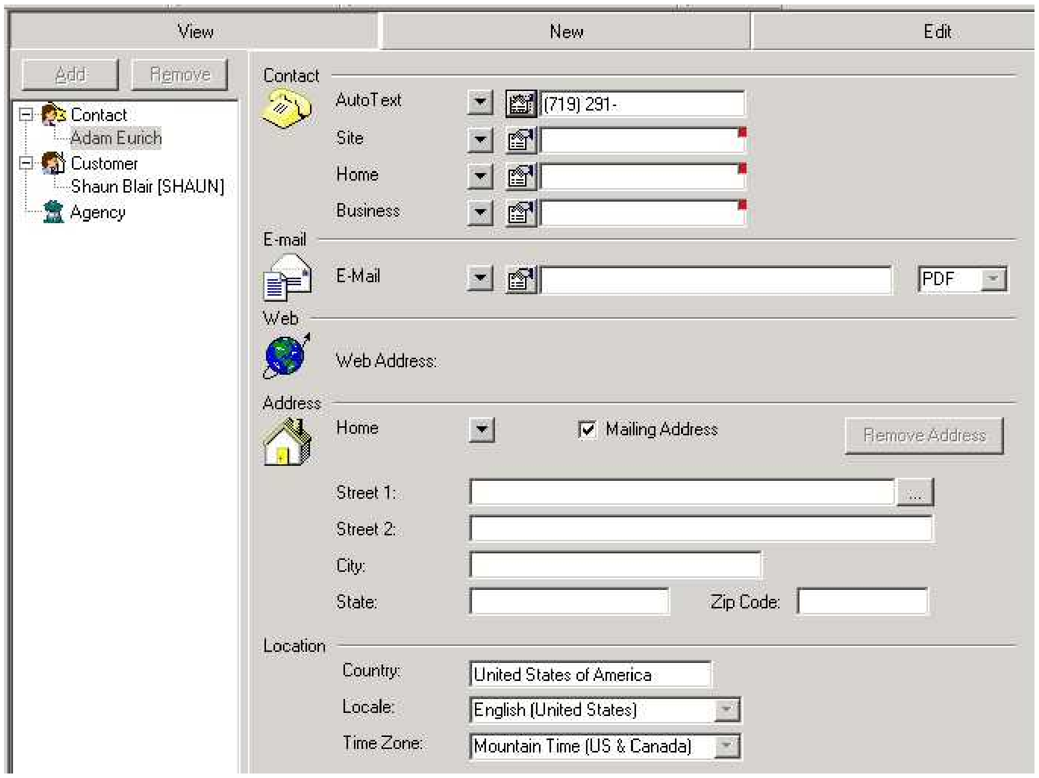
Perform the following steps to designate a Contact Point phone number:

1. Open the Operator Workstation.
2. Open the record for the Customer for whom you want to use AutoText.
3. Select the "Contact List" option from the Jump To menu.
4. Click the "Contact Details" icon.

Note: this icon is located to the left of the Jump To menu, and is represented by a house as shown in the following screenshot:



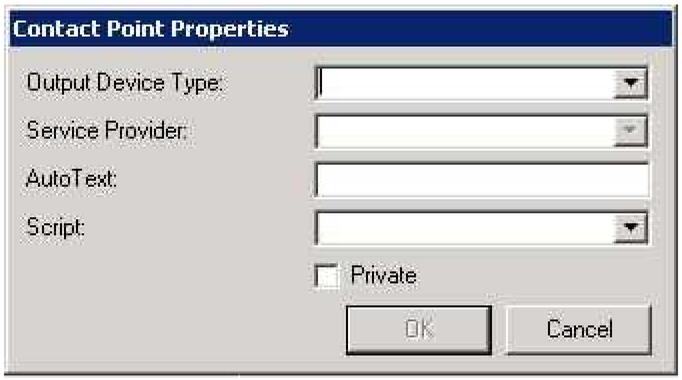
Result: the "Contact" form displays as shown in the following screenshot:



AutoText: (719) 291-
E-Mail format: PDF
Mailing Address: ✓
Country: United States of America
Locale: English (United States)
Time Zone: Mountain Time (US & Canada)

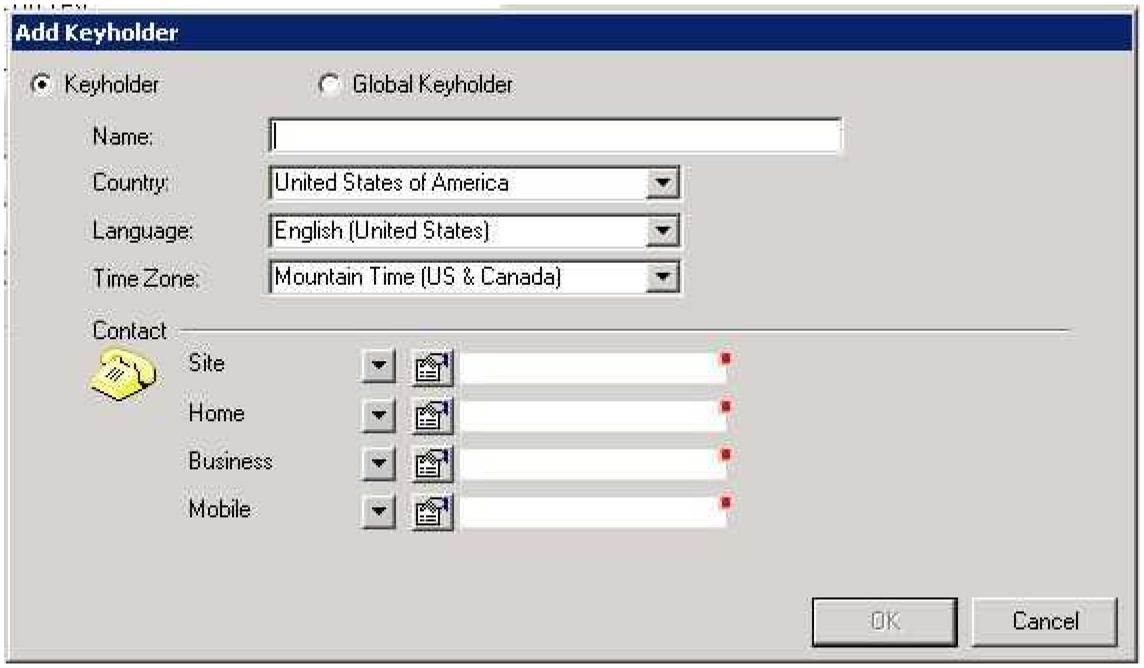
5. If the phone number you want to add is for an existing contact, click the contact's name in the Navigation Tree. Then, click the AutoText "Contact Point Properties" icon as shown in the following screenshot:

Result: the "Contact Point Properties" window displays as shown in the following screenshot:



6. If the phone number you want to add is for an existing contact, skip to step 13.
7. If the phone number you want to add is for a new contact, click "**Edit**" and then click "**Add**".

Result: the "Add Keyholder" window displays as shown in the following screenshot:


Add Keyholder

(● Keyholder) (○ Global Keyholder)

Name:

Country: United States of America

Language: English (United States)

Time Zone: Mountain Time (US & Canada)

Contact

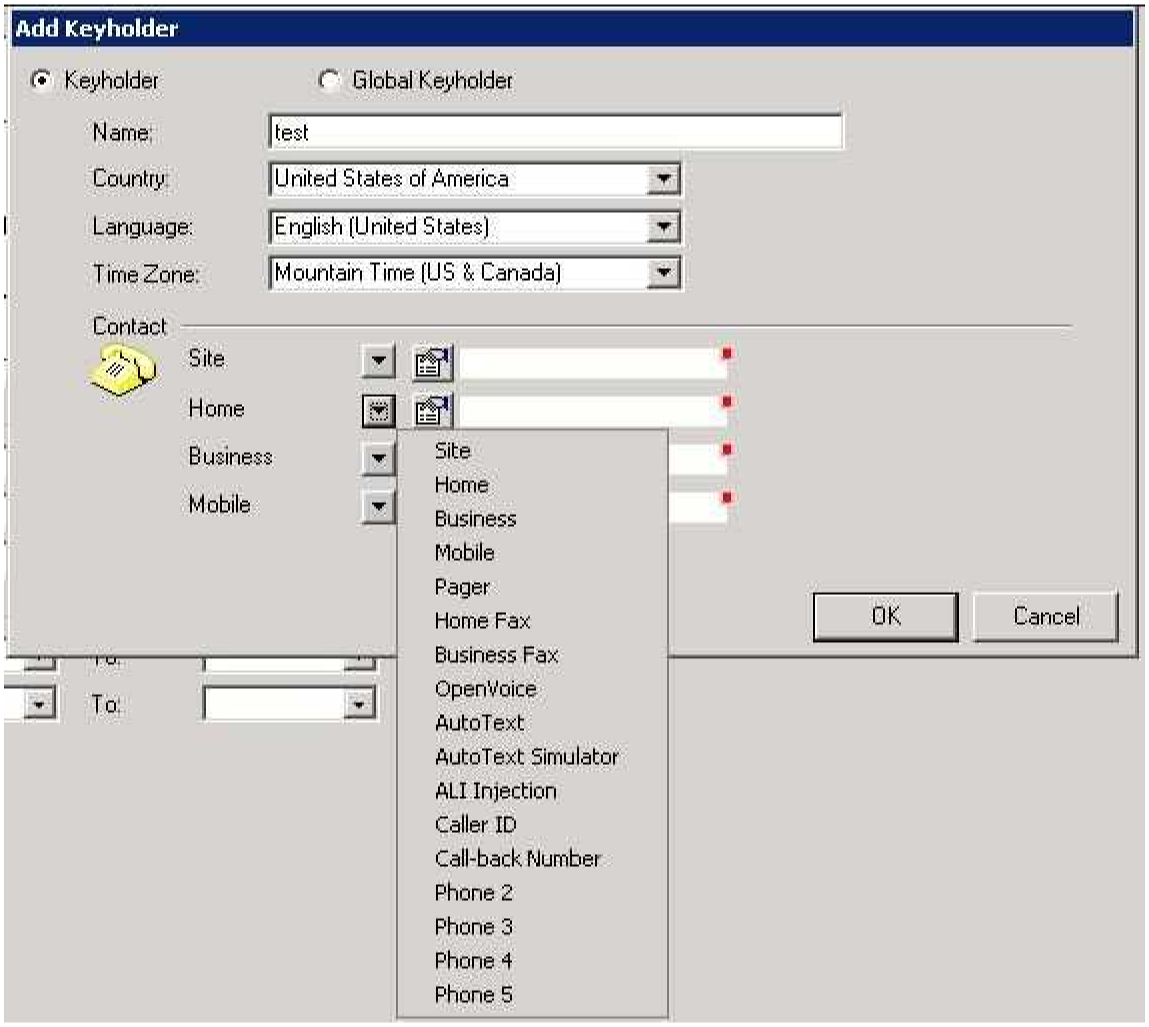
Site

Home

Business

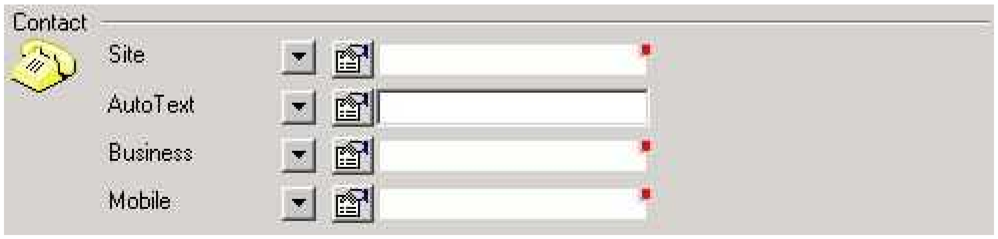
Mobile

8. Enter the contact's name and other relevant information.
9. In the "Contact" area of the window, enter the phone numbers you want for the contact in the available fields.
10. Select a "Contact" area field for which you have not already entered a phone number, and then click the down arrow to the left of the field as shown in the following screenshot:



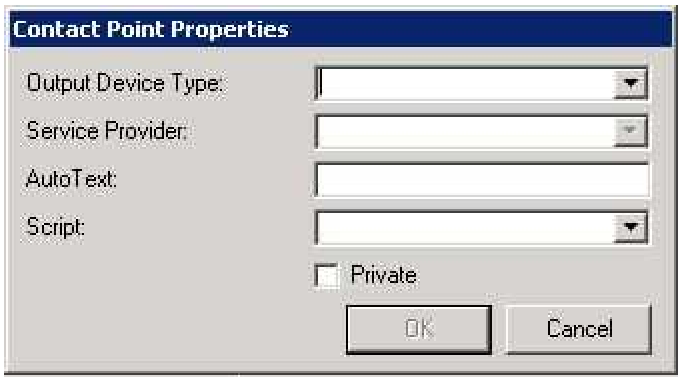
Name: test
Country: United States of America
Language: English (United States)
Time Zone: Mountain Time (US & Canada)

11. Select "AutoText".
Result: AutoText now displays as the label for the "Contact" area field you selected as shown in the following screenshot:



12. Click the AutoText "Contact Point Properties" icon as shown in the following screenshot:

Result: the "Contact Point Properties" window displays as shown in the following screenshot:



13. Select "MediaGateway" from the "Output Device Type:" dropdown menu.
14. Select "AutoText" from the "Service Provider:" dropdown menu.
15. Enter the phone number for the new contact into the "AutoText" field.
16. Select the script you recently created from the "Script:" dropdown menu.
17. If you want the contact information to remain private, select the "Private" checkbox.
18. Click "**OK"**.

Result: the "Contact Point Properties" window closes, and the system returns you to the "Contact" form.

19. Click "**Save**".

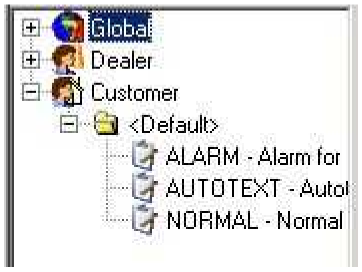
Adding the Script to an Action Pattern

Now that you have created a script and designated a Contact Point, you must include the script as part of an Action Pattern:

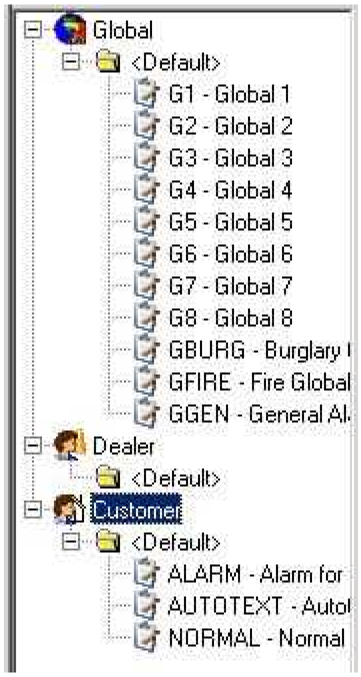
Perform the following steps to create an Action Pattern and include your script:

1. Open the Operator Workstation.
2. Open the record for the Customer for whom you want to use AutoText.
3. Click "Action Patterns" from the Jump To menu.

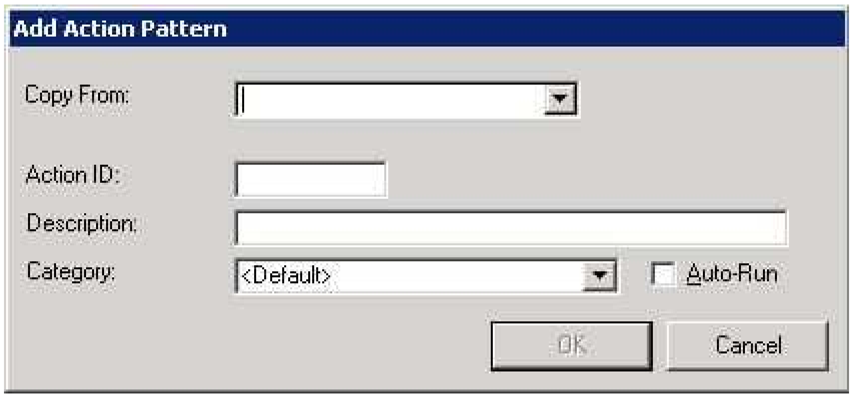
Result: the Action Pattern Navigation Pane displays as shown in the following screenshot:



4. Select the "Customer" node as shown in the following screenshot:



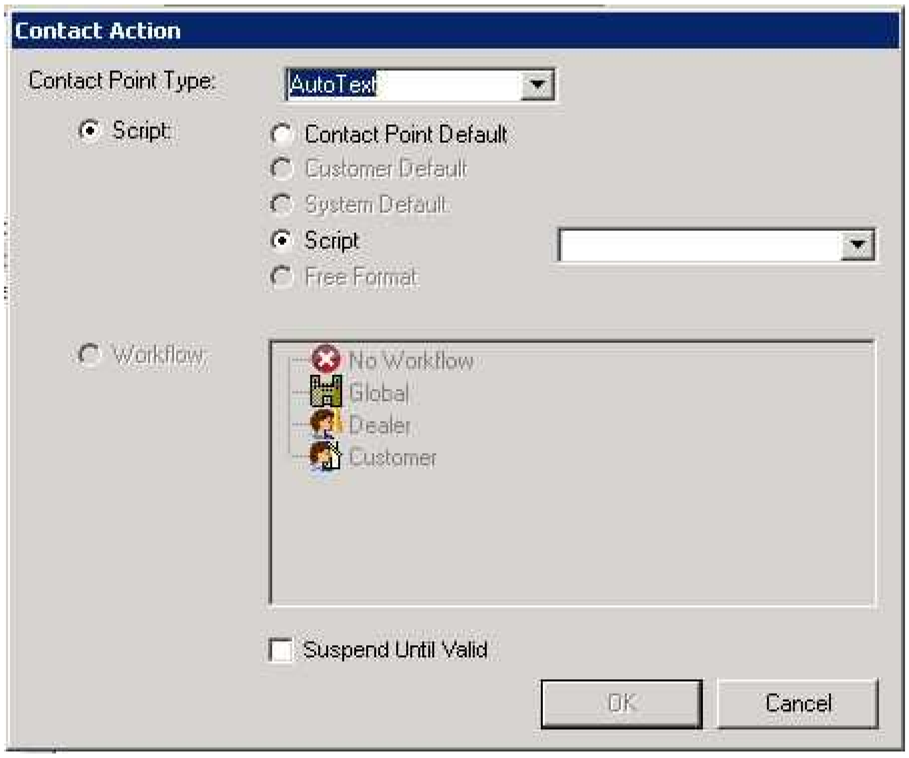
5. Click "**Edit**", and then click "**Add**".
Result: the "Add Action Pattern" window displays as shown in the following screenshot:



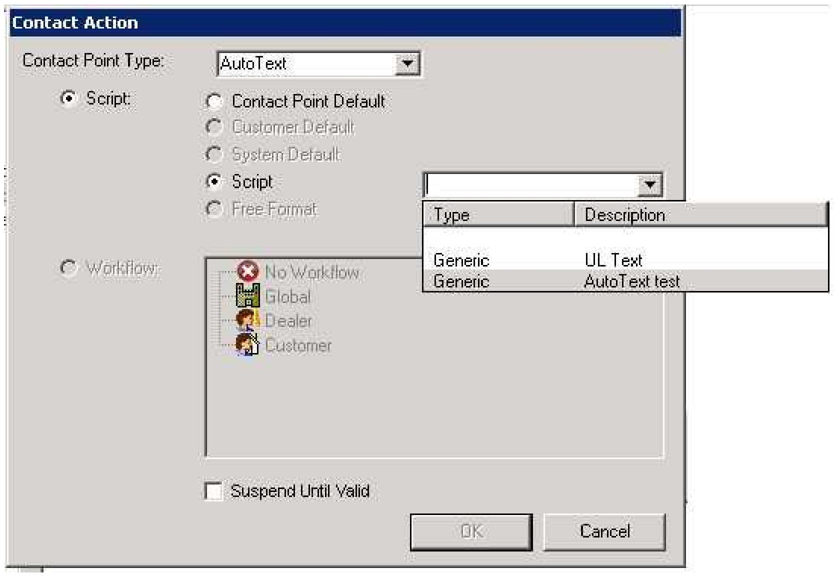
6. Enter an Action ID.
7. Enter a description.
8. Select a category from the "Category:" dropdown list.
9. If you want, select the "Auto-Run" checkbox, and then click "**OK**".
Note: having an Auto-Run designation is not necessary when you are operating in AutoClient mode.
10. Select the Action Pattern you just created from from the Navigation Tree.
11. Select the AutoText contact number you created for the Contact Point, and then click "

Add Command".

Result: the "Contact Action" window displays as shown in the following screenshot:



12. Select the script you created from the "Script" dropdown menu as shown in the following screenshot:



Type	Description
Generic	UL Text
Generic	AutoText test

13. If you want, select the "Suspend Until Valid" checkbox.
14. Click "**OK"**.

Result: the command you added now displays in the window below the "Add Command" button.
15. Click "**Save**".

Text Message Timeouts

When you use AutoText as part of an Action Pattern, Manitou suspends the alarm until it receives a response (and the Operator takes further action), or until it reaches a set timeout limit. Timeouts for AutoText can be set in the MediaGateway 2. These settings control how long the system waits for a response from the recipient before timing out and making the alarm active again.

AutoText has the following two default timeout settings (both are calculated in seconds):

- Initial Text Message Timeout (default time 1200 seconds = 20 minutes)
- Text Message Timeout (default time 120 seconds = 2 minutes)

Example of Timeouts in a Script:

The following example exchange is included to illustrate the intervals at which Manitou requires responsive communication:

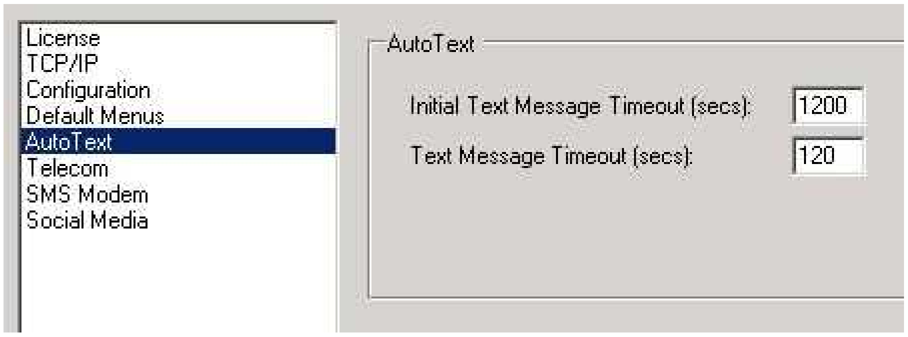
Initial message sent: *Alarm received. Would you like to cancel?* (Recipient has 20 minutes to respond.)
Response: *Yes*.
Message sent: *Please provide your username.* (Recipient has 2 minutes to respond.)
Response: *jhancock*
Message sent: *Please provide your password*. (Recipient has 2 minutes to respond.)
Response: *******

Modifying Text Message Timeouts

Perform the following steps to modify the default timeout interval settings:

1. Open the MediaGateway 2.
2. Click "Options" at the bottom of the Navigation Pane.
3. If the system prompts you for a password, enter your password and then click "**OK**".
4. Select "AutoText" from the "Options" Navigation Pane.

Result: the AutoText Options menu displays as shown in the following screenshot:



Initial Text Message Timeout (secs): 1200

Text Message Timeout (secs): 120

5. Modify the default values to the values you want.

Note: these fields accept any increment value of 4 characters or less.

6. Click "**Save**".

Completing the AutoText Setup

Perform the following steps to finalize the AutoText setup in the MediaGateway 2:

Adding an SMS Connector

If you have not already done so, you must set up the SMS Connector for the UniversalConnector. For instructions on setting up the SMS Connector, refer to the "SMS" document in the Connector Setup section.

Importing an AutoText Menu

If you have not already done so, you may find it necessary to import an AutoText menu into the MediaGateway 2. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Adding a Line Driver

If you have not already done so, you may find it necessary to add an AutoText Line Driver.

For instructions on adding a MediaGateway 2 Line Driver, refer to the "Line Drivers" document.
For instructions on using Parameters when setting up a Line Driver, refer to the "Parameters" document.

Note: if AutoText is not available as a MediaGateway 2 Line Function, you will need to import the AutoText menu. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Designating a Default Menu

If you have not already done so, you must designate a default menu for AutoText to use if no other menus have been designated.

For instructions on designating default menus, refer to the "Default Menu" document.

ChannelBank

ChannelBank answers signals from the MediaGateway 2, and then routes those calls to the Receiver. Receivers can be connected directly to the station ports of the MediaGateway 2. ChannelBank passes DNIS/Caller ID, and can detect certain line conditions (for e.g., no answer). It can also send an alarm into Manitou while automatically busying out the line.

ChannelBank scenario:

The following example is included to illustrate a common ChannelBank scenario:

A Receiver Line card has gone into a non-answering state at the same time a new call is received into the MediaGateway 2 ChannelBank. The ChannelBank routes the call from the PRI to the analog station card, but the Receiver does not answer. The MediaGateway 2 ChannelBank allows 15 seconds for the call to be answered, then it sets the channel to busy and sends a Line Fault alarm directly into Manitou. The next call coming into the line rings busy, and the hunt group moves it automatically onto the next line. An Operator receives the Line Fault alarm, and a technician examines the Receiver problem.

Setting up the ChannelBank in the MediaGateway 2

ChannelBank is a simple product that only requires set up in the MediaGateway 2.

Importing a ChannelBank Menu

If you have not already done so, you may find it necessary to import a ChannelBank menu into the MediaGateway 2. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Adding a ChannelBank Line Driver

If you have not already done so, you may find it necessary to add a ChannelBank Line Driver. For instructions on adding a MediaGateway 2 Line Driver,refer to the "Line Drivers" document.

For instructions on using Parameters when setting up a Line Driver, refer to the "Parameters" document.

Note: if ChannelBank is not available as a MediaGateway 2 Line Function, you will need to import it. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Designating a Default Menu

If you have not already done so, you must designate a default menu for ChannelBank to use if no other menus have been designated.

For instructions on designating default menus, refer to the "Default Menu" document.

LineCheck

The LineCheck component tests Receiver Lines and reports any failures.

Setting Up LineCheck in Manitou

You must create a Manitou account for the LineCheck component. The account must have the same Transmitter ID as the LineCheck menu items in the MediaGateway 2.

After you have created the LineCheck account, create an Action Pattern for the account that includes the following:
- set *DO to come in as a signal with no action required
- set *DF to come in as an alarm with action required. Then, apply the Action Pattern to perform the action.

Example:
- Notify IT

- Notify Manager
- Have Operator call phone lines to double-check

For more information on setting up accounts, refer to the appropriate manual for your Manitou system.

Completing the Setup in the MediaGateway 2

You must define LineCheck numbers in the MediaGateway 2 Line Drivers form.

Importing a LineCheck Menu

If you have not already done so, you may find it necessary to import a LineCheck menu into the MediaGateway 2. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Adding a LineCheck Line Driver

If you have not already done so, you must add a LineCheck Line Driver.
For instructions on adding a MediaGateway 2 Line Driver, refer to the "Line Drivers" document.
For instructions on using Parameters when setting up a Line Driver, refer to the "Parameters" document.

Note: if Line Check is not available as a MediaGateway 2 Line Function, you will need to import it. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

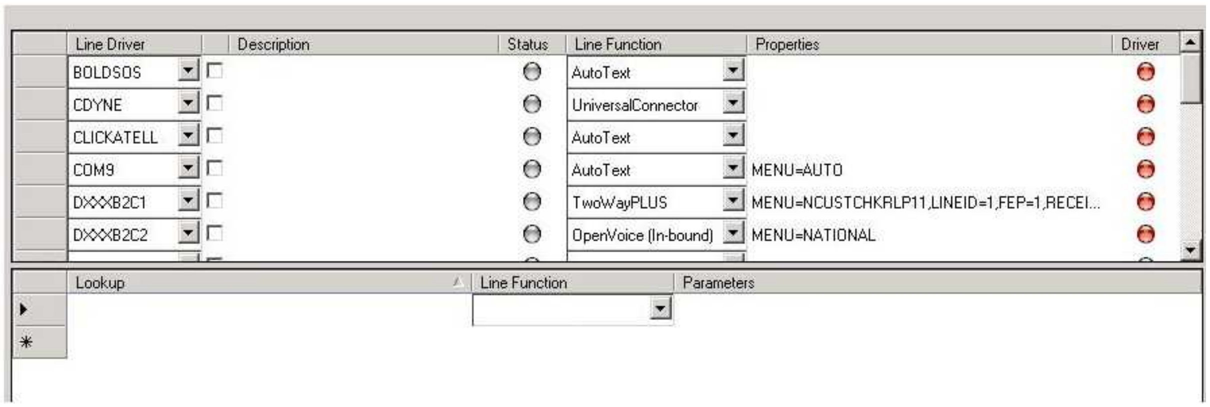
Adding Lines to the Lookup

After you have set up the Line Driver, you must add the phone numbers you want to the Line Driver Lookup fields so the system knows which numbers to dial.

Perform the following steps to include the phone numbers you want in the Line Driver Lookup fields:

1. Open the MediaGateway 2.
2. Select "Line Driver" from the Navigation Tree under Administration.

Result: the Line Driver form displays as shown in the following screenshot:

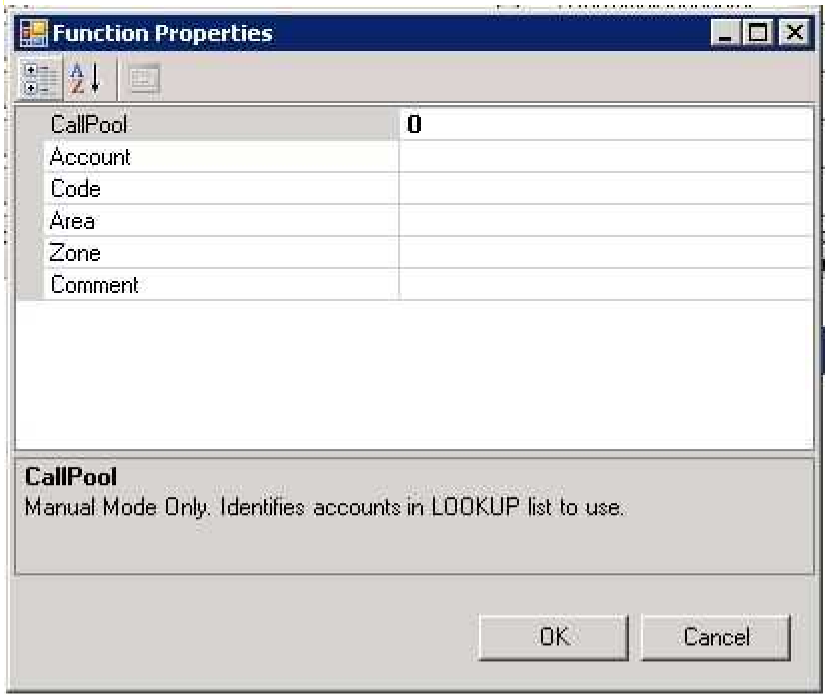


	Line Driver	Description	Status	Line Function	Properties	Driver
	BOLDSOS	☐	●	AutoText		●
	CDYNE	☐	●	UniversalConnector		●
	CLICKATELL	☐	●	AutoText		●
	COM9	☐	●	AutoText	MENU=AUTO	●
	DXXXB2C1	☐	●	TwoWayPLUS	MENU=NCUSTCHKRLP11,LINEID=1,FEP=1,RECEI...	●
	DXXXB2C2	☐	●	OpenVoice (In-bound)	MENU=NATIONAL	●

	Lookup	Line Function	Parameters
▶			
*			

3. Click in the first available Lookup field, and enter the 10-digit phone number you want to add to the Lookup.
4. Select "LineCheck" from the Line Function pulldown menu.
5. Double-click in the Parameters area of the Window for the current Lookup row.
6. Right-click in the Parameters area, and select "Properties".

Result: the Properties Function window displays as shown in the following screenshot:



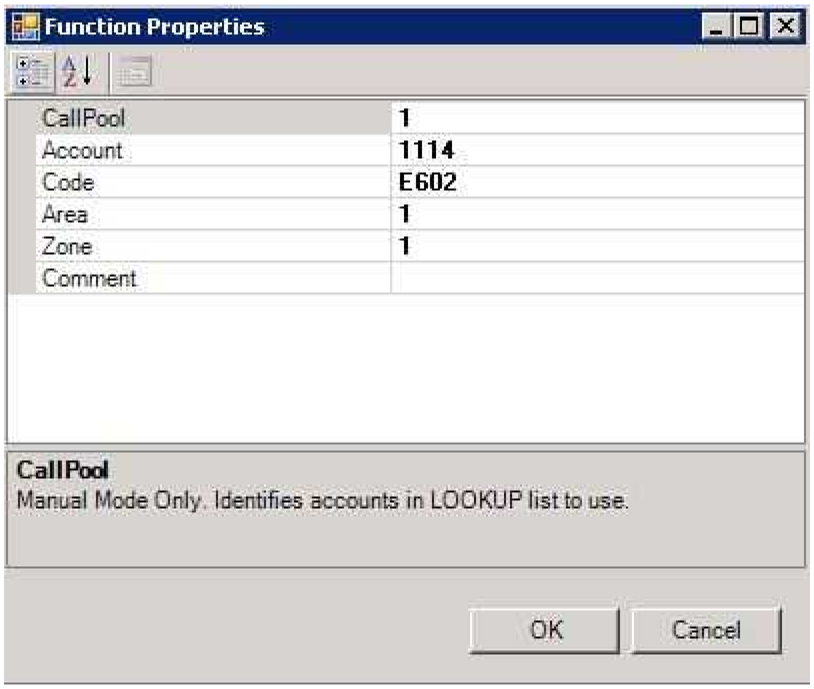
CallPool	0
Account	
Code	
Area	
Zone	
Comment	

CallPool
Manual Mode Only. Identifies accounts in LOOKUP list to use.

7. Populate the following fields in the Function Properties window:
   - **CallPool** - lines that are assigned to a call pool to balance the load
   - **Account** - account number to send using Contact ID
   - **Code** - Contact ID
   - **Area** - area to send using Contact ID
   - **Zone** - zone to send using Contact ID

- **Comment** - optional comment for this phone number

Note: the following screenshot is provided as an example only. The values you enter for these fields may be different than those displayed.

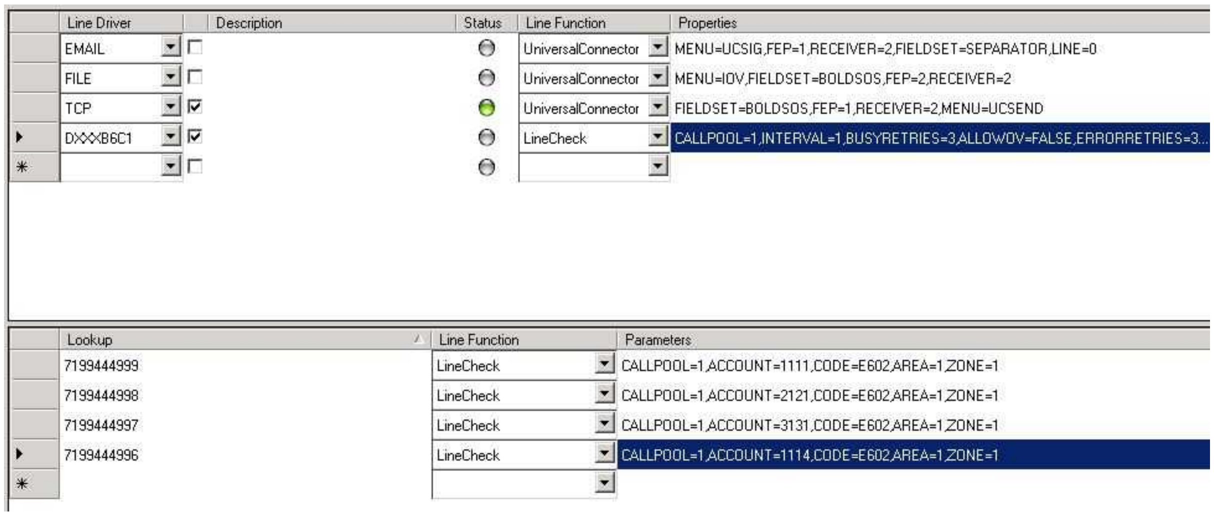


CallPool	1
Account	1114
Code	E602
Area	1
Zone	1
Comment	

CallPool
Manual Mode Only. Identifies accounts in LOOKUP list to use.

LineCheck Parameters

8. When you finish entering information into the Function Properties window, click "**OK**".



	Line Driver		Description	Status	Line Function	Properties
	EMAIL	☐		○	UniversalConnector	MENU=UCSIG,FEP=1,RECEIVER=2,FIELDSET=SEPARATOR,LINE=0
	FILE	☐		○	UniversalConnector	MENU=IOV,FIELDSET=BOLDSOS,FEP=2,RECEIVER=2
	TCP	☑		●	UniversalConnector	FIELDSET=BOLDSOS,FEP=1,RECEIVER=2,MENU=UCSEND
▸	DXXXB6C1	☑		○	LineCheck	CALLPOOL=1,INTERVAL=1,BUSYRETRIES=3,ALLOWOV=FALSE,ERRORRETRIES=3...
*		☐		○		

	Lookup	Line Function	Parameters
	7199444999	LineCheck	CALLPOOL=1,ACCOUNT=1111,CODE=E602,AREA=1,ZONE=1
	7199444998	LineCheck	CALLPOOL=1,ACCOUNT=2121,CODE=E602,AREA=1,ZONE=1
	7199444997	LineCheck	CALLPOOL=1,ACCOUNT=3131,CODE=E602,AREA=1,ZONE=1
▸	7199444996	LineCheck	CALLPOOL=1,ACCOUNT=1114,CODE=E602,AREA=1,ZONE=1
*			

Sample LineCheck setup

9. Click "**Save**".

Designating a Default Menu

You must designate a default menu for LineCheck to use if no other menus have been designated.

For instructions on designating default menus, refer to the "Default Menu" document.

OpenVoice

OpenVoice uses text-to-speech technology to provide responses for both inbound and outbound voice capabilities.

- Outbound OpenVoice: allows outbound, automated voice calls that are interactive. The calls are driven by Manitou Action Patterns and a user-definable scripting language (HAL - Hierarchical Application Language).
- Inbound OpenVoice: allows inbound calls to an IVR (Interactive Voice Response) system. It can be used for technician level functions, guard level functions, check-ins, or Customer requests. It is driven by a user-definable scripting language (HAL - Hierarchical Application Language).

Inbound OpenVoice

Inbound OpenVoice provides incoming callers with automated, menu-based phone dialog. Because the Inbound OpenVoice function does not require a script, setup is only required in the MediaGateway 2.

Importing an Inbound OpenVoice Menu

You must import an Inbound OpenVoice menu into the MediaGateway 2. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Adding a Line Driver

Next, you must add an Inbound Voice Line Driver.
For instructions on adding a MediaGateway 2 Line Driver, refer to the "Line Drivers" document.
For instructions on using Parameters when setting up a Line Driver, refer to the "Parameters" document.

Note: if Inbound OpenVoice is not available as a MediaGateway 2 Line Function, you will need to import it. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Designating a Default Menu

If you have not already done so, you must designate a default menu for Inbound OpenVoice to use if no other menus have been designated.

For instructions on designating default menus, refer to "Default Menus" dialog.

Outbound OpenVoice

Outbound OpenVoice provides automated, interactive voice calls using Manitou Action Patterns and HAL (Hierarchical Application Language). You can use Outbound OpenVoice to automate outbound calls on low-priority or action-type alarms by giving the Auto-Client the ability to communicate with the user.

Setting up Outbound OpenVoice in the MediaGateway 2:

You must configure Outbound OpenVoice in the MediaGateway 2.

Importing an Outbound OpenVoice Menu

You must import an Outbound Open Voice menu into the MediaGateway 2. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Adding a Line Driver

Next, you must add an Outbound OpenVoice Line Driver.

For instructions on adding a MediaGateway 2 Line Driver, refer to the "Line Drivers" document.
For instructions on using Parameters when setting up a Line Driver, refer to the "Parameters" document.

Note: if Outbound OpenVoice is not available as a MediaGateway 2 Line Function, you will need to import it. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Designating a Default Menu

If you have not already done so, you must designate a default menu for Outbound OpenVoice to use if no other menus have been designated.

For instructions on designating default menus, refer to the "Default Menus" document.

Setting up Outbound OpenVoice in Manitou

You must configure Outbound OpenVoice in Manitou.

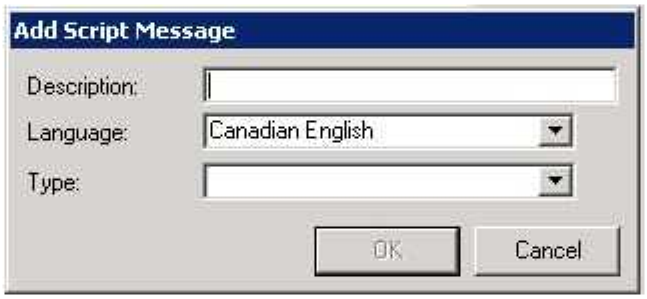
Creating the Script

Scripts are created and managed in the Manitou Supervisor Workstation.

Perform the following steps to add a script:

1. Open the Supervisor Workstation
2. Navigate to the Maintenance menu, and then select "Script Messages".
3. Click "**Edit**", and then click "**Add**".

Result: the "Add Script Message" displays as shown in the following screenshot:


Add Script Message

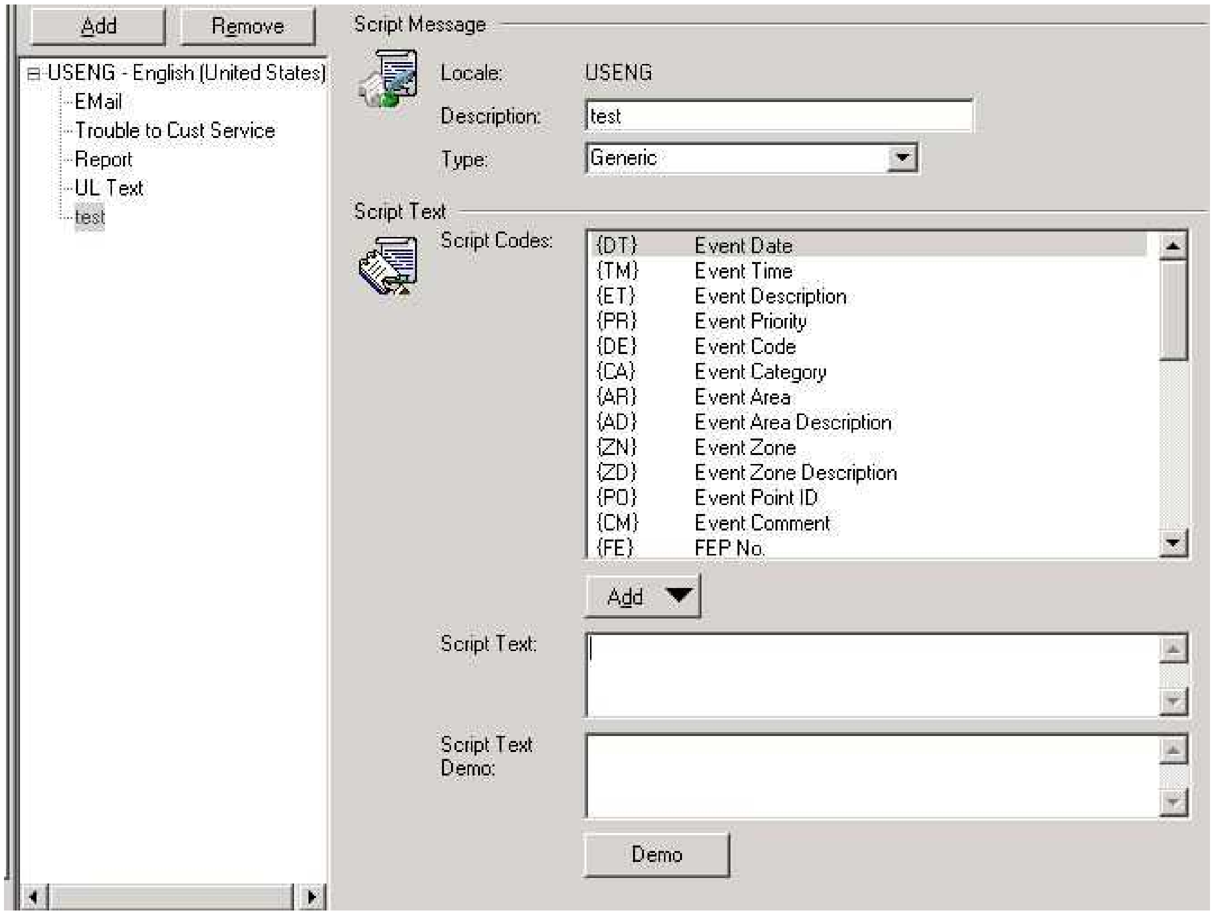
Description:

Language: Canadian English

Type:

4. Enter a description, and select a language from the "Language:" dropdown menu.
5. Select "Generic" from the "Type:" dropdown menu, and then click "**OK**".

Result: the "Add Script Message" window closes and the "Script Message" window displays as shown in the following screenshot:



USENG - English (United States)
- EMail
- Trouble to Cust Service
- Report
- UL Text
- test

Script Message

Locale: USENG

Description: test

Type: Generic

Script Text

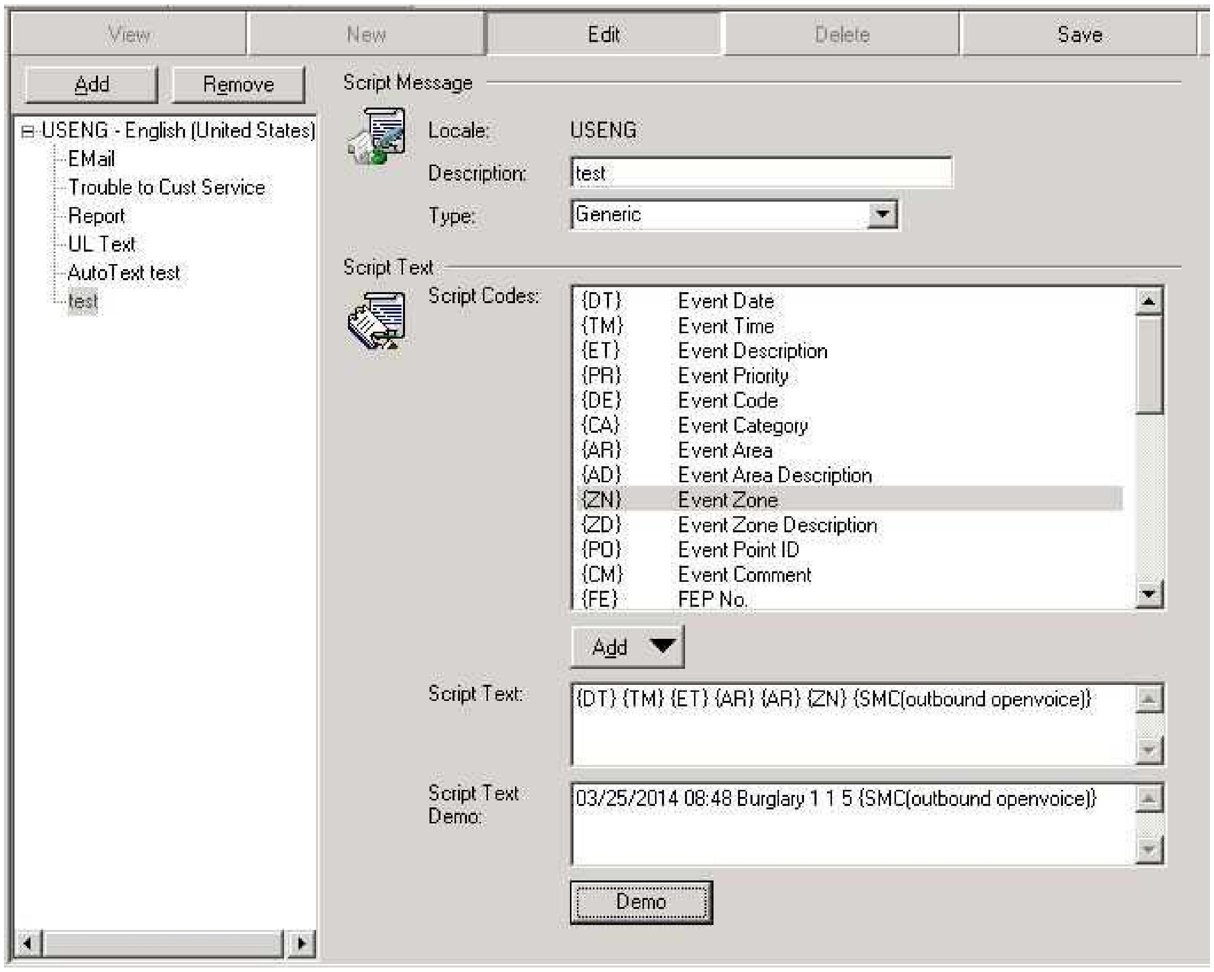
Script Codes:

{DT}	Event Date
{TM}	Event Time
{ET}	Event Description
{PR}	Event Priority
{DE}	Event Code
{CA}	Event Category
{AR}	Event Area
{AD}	Event Area Description
{ZN}	Event Zone
{ZD}	Event Zone Description
{PO}	Event Point ID
{CM}	Event Comment
{FE}	FEP No.

6. Select the codes you want from the "Script Codes:" window, and then add them to the "Script Text:" window using the "**Add**" button.
7. Enter the text for your script message into the "Script Text:" window.
8. Enter the following code at the end of your script message text: "{SMC(menu name)}". Insert the actual MediaGateway 2 menu name into the code.

Note: this ensures that the MediaGateway 2 is engaged for the Outbound OpenVoice feature.

9. If you want, click "**Demo**" to show how the script message will display to your recipient.



Locale: USENG

Description: test

Type: Generic

Script Codes:

{DT}	Event Date
{TM}	Event Time
{ET}	Event Description
{PR}	Event Priority
{DE}	Event Code
{CA}	Event Category
{AR}	Event Area
{AD}	Event Area Description
{ZN}	Event Zone
{ZD}	Event Zone Description
{PO}	Event Point ID
{CM}	Event Comment
{FE}	FEP No.

Script Text: {DT} {TM} {ET} {AR} {AR} {ZN} {SMC(outbound openvoice)}

Script Text Demo: 03/25/2014 08:48 Burglary 1 1 5 {SMC(outbound openvoice)}

10. Click "**Save**".

Applying the Script

Once you have created a script in the Supervisor Workstation, you must designate the Contact Point phone number for which you want to use Outbound OpenVoice in the Operator Workstation.

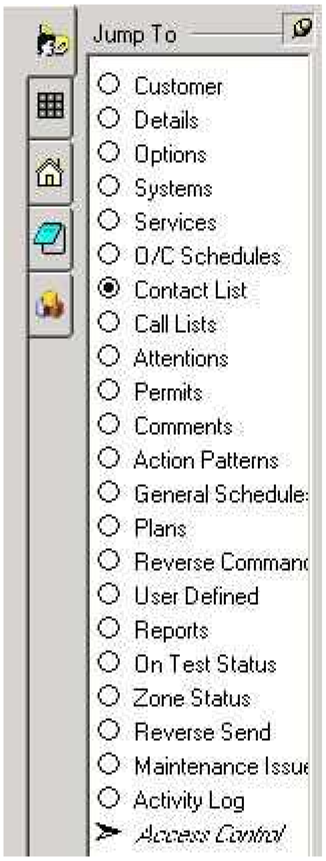
Designating the Contact Point Phone Number

Perform the following steps to designate a Contact Point phone number:

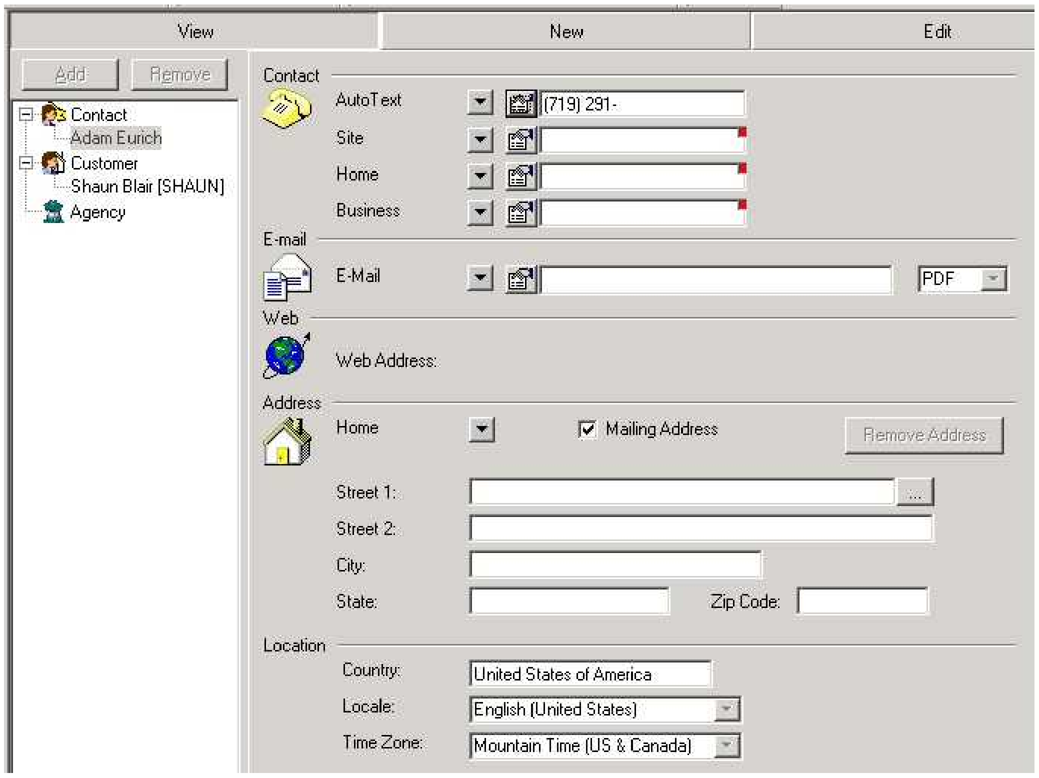
1. Open the Operator Workstation.

2. Open the record for the Customer for whom you want to use Outbound OpenVoice.
3. Select the "Contact List" option from the Jump To menu.
4. Click the "Contact Details" icon.

Note: this icon is located to the left of the Jump To menu, and is represented by a house as shown in the following screenshot:



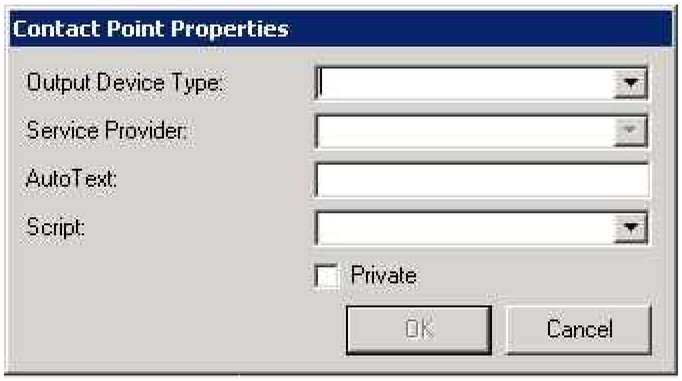
Result: the "Contact" form displays as shown in the following screenshot:



AutoText: (719) 291-
Mailing Address: ✓
Country: United States of America
Locale: English (United States)
Time Zone: Mountain Time (US & Canada)

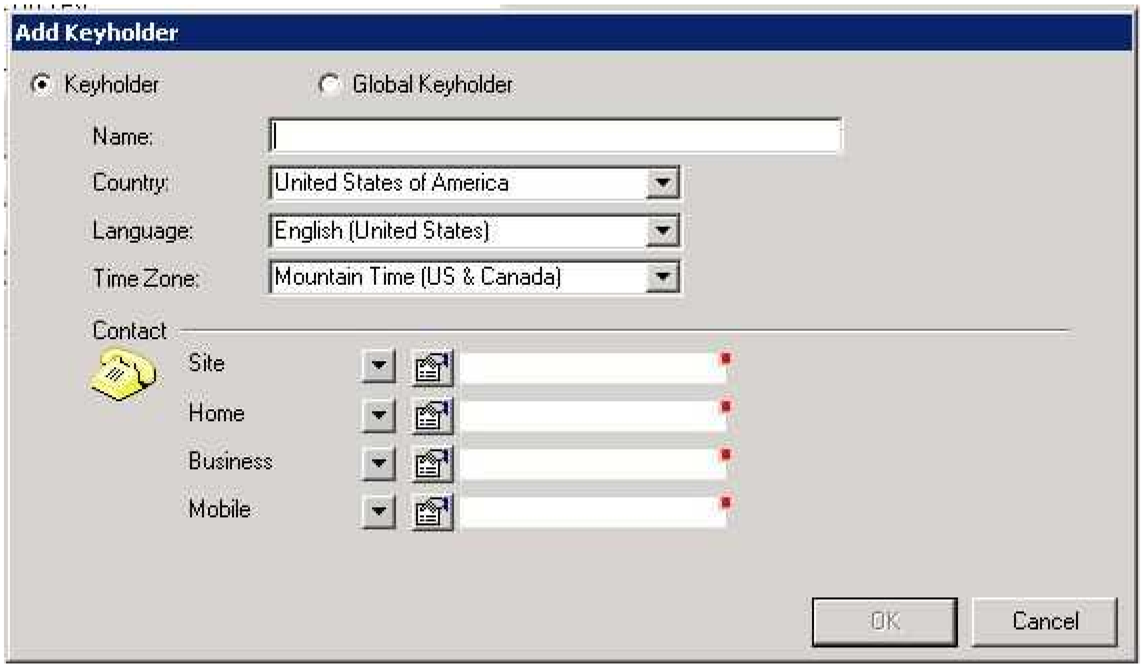
5. If the phone number you want to add is for an existing contact, click the contact's name in the Navigation Tree. Then, click the "Contact Point Properties" icon for the phone line you want to use for Outbound OpenVoice as shown in the following screenshot:

Result: the "Contact Point Properties" window displays as shown in the following screenshot:



6. If the phone number you want to add is for an existing contact, skip to step 13.
7. If the phone number you want to add is for a new contact, click "**Edit**" and then click "**Add**".

Result: the "Add Keyholder" window displays as shown in the following screenshot:


Add Keyholder

[✓ Keyholder] Global Keyholder

Name:	
Country:	United States of America
Language:	English (United States)
Time Zone:	Mountain Time (US & Canada)

Contact

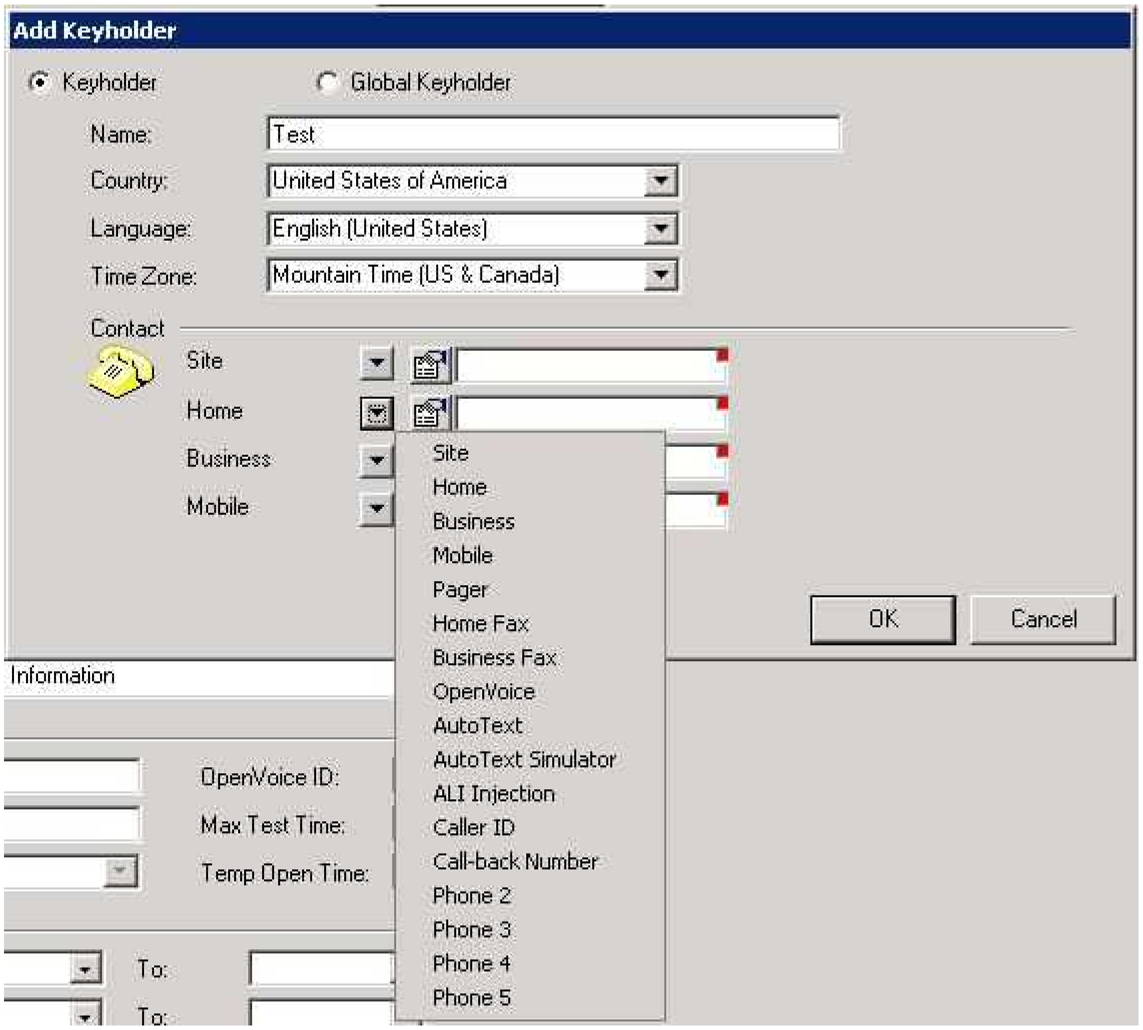
Site

Home

Business

Mobile

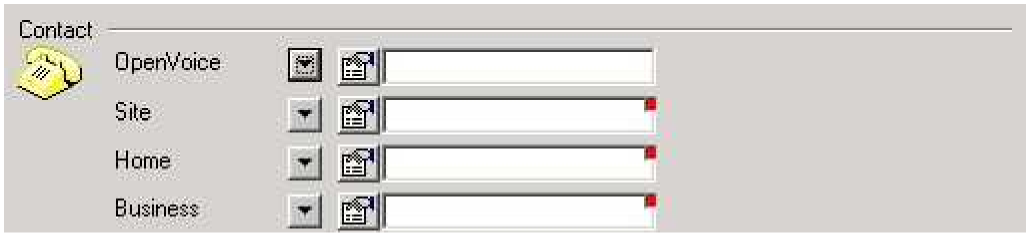
8. Enter the contact's name and other relevant information.
9. In the "Contact" area of the window, enter the phone numbers you want for the contact in the available fields.
10. Select a "Contact" area field for which you have not already entered a phone number, and click the down arrow to the left of the field as shown in the following screenshot:


Add Keyholder

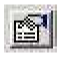
(● Keyholder) ○ Global Keyholder

Name: Test
Country: United States of America
Language: English (United States)
Time Zone: Mountain Time (US & Canada)

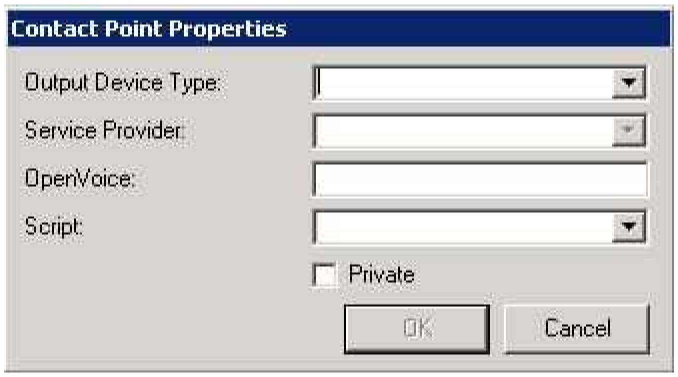
11. Select "OpenVoice".
Result: OpenVoice now displays as the label for the "Contact" area field you selected as shown in the following screenshot:



12. Click the OpenVoice "Contact Point Properties" icon as shown in the following screenshot:



Result: the "Contact Point Properties" window displays as shown in the following screenshot:



13. Select MediaGateway from the "Output Device Type:" dropdown menu.
14. Select "OpenVoice" from the "Service Provider:" dropdown menu.
15. Enter the phone number for the new contact into the "OpenVoice:" field.
16. Select the script you recently created from the "Script:" dropdown menu.
17. If you want the contact information to remain private, select the "Private" checkbox.
18. Click "**OK"**.

Result: the "Contact Point Properties" window closes, and the system returns you to the "Contact" form.

19. Click "**Save**".

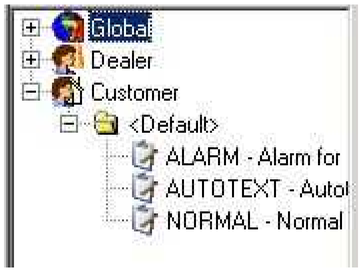
Adding the Script to an Action Pattern

Now that you have created a script and designated a Contact Point, you must include the script as part of an Action Pattern:

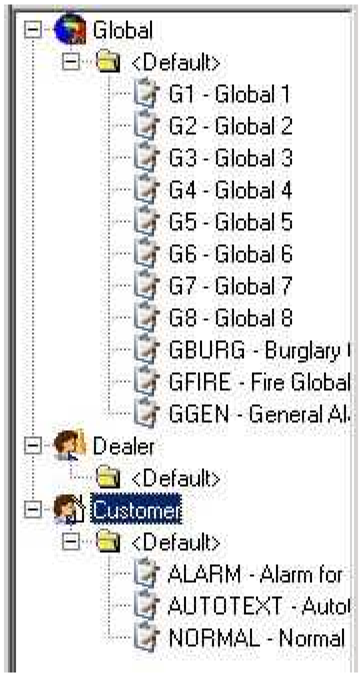
Perform the following steps to create an Action Pattern and include your script:

1. Open the Operator Workstation.
2. Open the record for the Customer for whom you want to use OpenVoice.
3. Click "Action Patterns" from the Jump To menu.

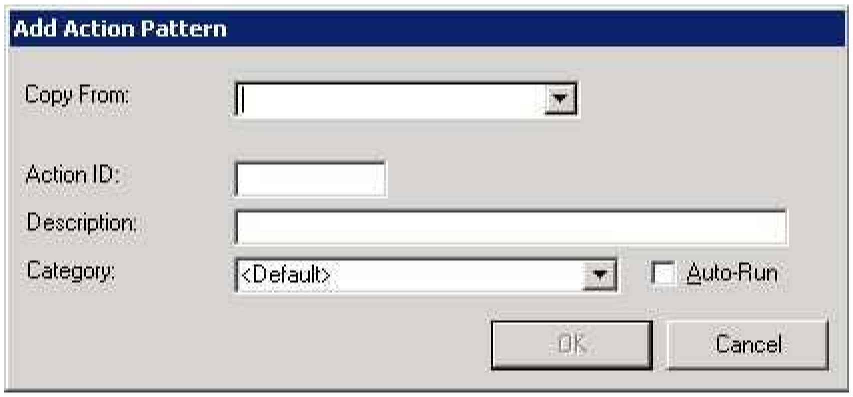
Result: the Action Pattern Navigation Pane displays as shown in the following screenshot:



4. Select the "Customer" node as shown in the following screenshot:



5. Click "**Edit**", and then click "**Add**".
Result: the "Add Action Pattern" window displays as shown in the following screenshot:



6. Enter an Action ID.

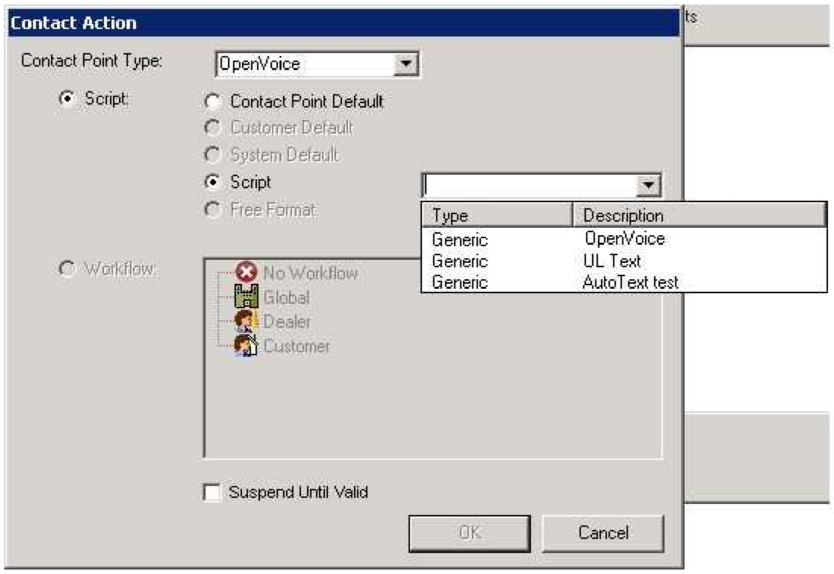
7. Enter a description.
8. Select a category from the "Category:" dropdown list.
9. If you want, select the "Auto-Run" checkbox, and then click "**OK**".

Note: having an Auto-Run designation is not necessary when you are operating in AutoClient mode.

10. Select the Action Pattern you just created from the Navigation Tree.
11. Select the OpenVoice contact number you created for the Contact Point, and then click " **Add Command**".

Result: the Contact Action window displays as shown in the following screenshot:

12. Select the script you created from the "Script" dropdown menu as shown in the following screenshot:



Contact Action

Contact Point Type: OpenVoice

[✓ Script:]

Contact Point Default
Customer Default
System Default
[✓ Script]
Free Format

Type	Description
Generic	OpenVoice
Generic	UL Text
Generic	AutoText test

Workflow:

No Workflow
Global
Dealer
Customer

Suspend Until Valid

13. If you want, select the "Suspend Until Valid" checkbox.
14. Click "**OK"**.

Result: the command you added now displays in the window below the "Add Command" button.

15. Click "**Save**".

The Loquendo Process

Manitou uses Loquendo Voices to provide lifelike voices to automated voice systems.

Deploying Loquendo Voices

Prior to installation, the Bold Project Manager will order Loquendo Voices for you. The Bold Project Manager will then send an email to the assigned technician with the necessary download information for the installation.

Note: each installation includes unique login information.

Perform the following steps to deploy Loquendo Voices:

1. Log into the MicroAutomation website, and then download all the Loquendo Voices installation files to the same machine.

Note: this eliminates the need for logging into the MicroAutomation website from each machine.

2. Download the complete folder for each PIK (license number), and then save them to your hard drive.

Note: Bold will provide you a PIK (license number) for each server you install.

3. Copy the files for each PIK to each, individual machine.
4. Each folder contains at least the following three .zip files:
   - Loquendo_TTS_7-Win32_Engine_Full_Distribution_7.9.8.zip
   - Loquendo_TTS_7-Win32_English_Distribution_7.8.2.zip
   - Loquendo_TTS_7-Win32_<character>_High_Quality_Distribution_7.7.0.zip ("<character>" represents the name of each, individual voice)
5. Unzip and install the above files in the order in which they are listed.

Loading Files to Individual Machines

Perform the following steps to load the Loquendo Voices files to individual machines:

1. Run the Loquendo License Manager files on each machine.
2. Enter the PIK for each machine and an email address into the appropriate fields.
3. If necessary, choose a network adapter from the dropdown menu.
4. When prompted, click "**I need to obtain a license for this computer**".
5. Save the license request text files.
6. Rename the license request files with unique names to indicate each machine you have licensed.
7. Collect and email all the text files from the license requests to Kevin Scott (scott@microautomation.com). Provide the necessary Customer information and

whether any of the requested licenses are intended to be for primary and secondary machines.

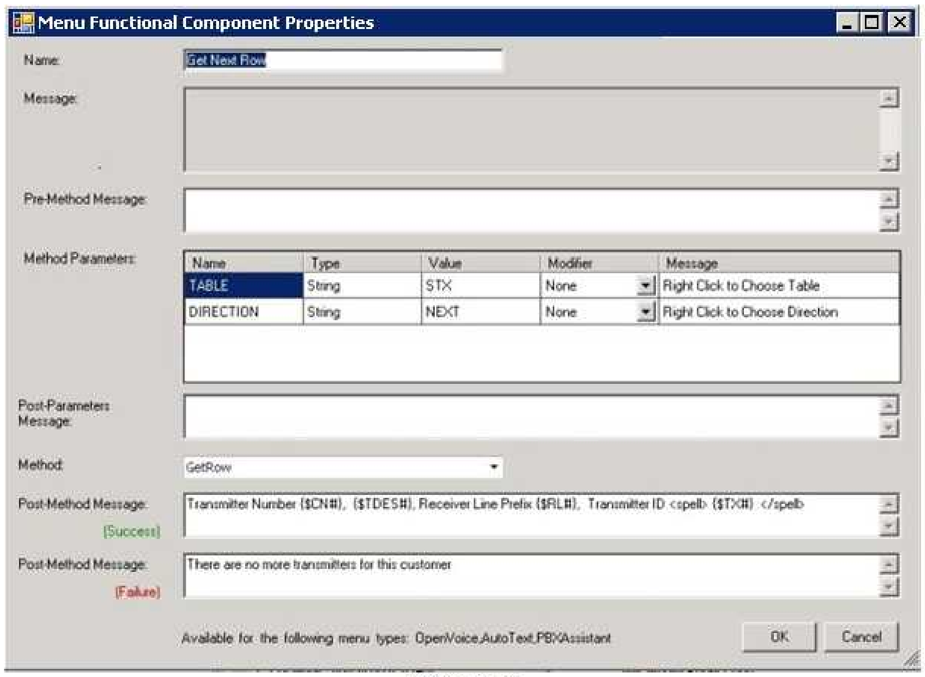
8. Once you receive the requested license files, click "**Extract and Deploy Licenses**".

The installation and licensing of Loquendo Voice is now complete.

SAPI XML Tags

The MediaGateway 2 uses SAPI XML tags. These tags control various aspects of outbound speech.

SAPI XML tags are defined in the MediaGateway 2 in the "Menu Functional Components Properties" form. To use SAPI XML tags, navigate to the "Menu Functional Components Properties" form as displayed in the following screenshot:



Menu Functional Component Properties

Name: Get Next Row

Message:

Pre-Method Message:

Method Parameters:

Name	Type	Value	Modifier	Message
TABLE	String	STX	None	Right Click to Choose Table
DIRECTION	String	NEXT	None	Right Click to Choose Direction

Post-Parameters Message:

Method: GetRow

Post-Method Message: (Success) Transmitter Number {$CN#}, {$TDES#}, Receiver Line Prefix {$RL#}, Transmitter ID <spell> {$TX#} </spell>

Post-Method Message: (Failure) There are no more transmitters for this customer

Available for the following menu types: OpenVoice,AutoText,PBXAssistant

MFC Properties

Voice State Control Tags

SAPI TTS XML supports the following five tags: Volume, Rate, Pitch, Emph, and Spell. These tags control the state of the current voice.

Volume

The Volume tag controls the volume of a voice. If the tag is empty, it applies to all subsequent text. If it has its own content, it only applies to that content. The Volume tag has one required attribute: Level. The value of this attribute should be an integer between zero and one hundred. Values outside this range will be truncated.

```
<volume level="50">
This text should be spoken at volume level fifty.

 <volume level="100">
This text should be spoken at volume level one hundred.
 </volume>

</volume>

<volume level="80"/>
All text which follows should be spoken at volume level eighty.
```

One hundred represents the default volume of a voice. Lower values represent percentages of this default (i.e., 50 corresponds to 50% of full volume).
Values specified using the Volume tag will be combined with values specified programmatically (using ISpVoice::SetVolume). For example, if you combine a SetVolume(50) call with a <volume level="50"> tag, the volume of the voice should be 25% of its full volume.

Rate

The Rate tag controls the rate of a voice. If the tag is empty, it applies to all subsequent text. If it has its own content, it only applies to that content.
The Rate tag has two attributes: Speed and AbsSpeed. One of these attributes must be present. The value of both of these attributes should be an integer between negative ten and ten. Values outside of this range may be truncated by the engine (but are not truncated by SAPI). The AbsSpeed attribute controls the absolute rate of the voice. Therefore, a value of ten always corresponds to a value of ten.

```
<rate absspeed="5">
This text should be spoken at rate five.

<rate absspeed="-5">
This text should be spoken at rate negative five.
</rate>
</rate>
```

```
<rate absspeed="10"/>
```

All text that follows should be spoken at rate ten.

Speed

The Speed attribute controls the relative rate of the voice. The absolute value is found by adding each Speed to the current absolute value.

```
<rate speed="5">
This text should be spoken at rate five.
<rate speed="-5">
This text should be spoken at rate zero.
</rate>
</rate>
```

Zero represents the default rate of a voice. Positive values are faster and negative values are slower. Values specified using the Rate tag will be combined with values specified programmatically (using ISpVoice::SetRate).

Pitch

The Pitch tag controls the pitch of a voice. If the tag is empty, it applies to all subsequent text. If it has its own content, it only applies to that content.
The Pitch tag has the following two attributes: Middle and AbsMiddle. One of the attributes must be present. The value of both of these attributes should be an integer between negative ten and ten. Values outside of this range may be truncated by the engine (but are not truncated by SAPI).

The AbsMiddle attribute controls the absolute pitch of the voice. Therefore, a value of ten always corresponds to a value of ten.

```
<pitch absmiddle="5">
This text should be spoken at pitch five.
 <pitch absmiddle="-5">
This text should be spoken at pitch negative five.
</pitch>
</pitch>
<pitch absmiddle="10"/>
```

All text that follows should be spoken at pitch ten.
The Middle attribute controls the relative pitch of the voice. The absolute value is found by adding each Middle to the current absolute value.

```
<pitch middle="5">
This text should be spoken at pitch five.
<pitch middle="-5">
This text should be spoken at pitch zero.
</pitch>
</pitch>
```

Zero represents the default middle pitch for a voice, with positive values being higher and negative values being lower.

Emph

The Emph tag instructs the voice to emphasize a word or section of text. The Emph tag cannot be empty. The following word should be emphasized:

```
<emph> boo </emph>!
```

The method of emphasis may vary from voice to voice.

Spell

The Spell tag forces the voice to spell out all text, rather than using its default word and sentence breaking rules, normalization rules, etc. All characters should be expanded to corresponding words (including punctuation, numbers, etc.). The Spell tag cannot be empty.

```
<spell>
These words should be spelled out.
</spell>
These words should not be spelled out.
```

Direct item insertion tags

The following three tags are supported: Silence, Pron, and Bookmark. These tags give applications the ability to directly insert items at some levels.

Silence

The Silence tag inserts a specified number of milliseconds of silence into the output audio stream. This tag must be empty. It must have the Msec attribute.

```
Five hundred milliseconds of silence <silence msec="500"/> just
occurred.
```

Pron

The Pron tag inserts a specified pronunciation. The voice then processes the sequence of phonemes exactly as they are specified. This tag can be empty, or it can have content. If it has content, it is interpreted as providing the pronunciation for the enclosed text. If the tag has content, the system will not process the enclosed text as it normally would.

The Pron tag has the following attribute: Sym. The value is a string of white space separated phonemes.

```
<pron sym="h eh 1 l ow & w er 1 l d "/>
<pron sym="h eh 1 l ow & w er 1 l d"> hello world </pron>
```

Bookmark

The Bookmark tag inserts a bookmark event into the output audio stream. Use this event to signal the application when the audio corresponding to the text at the Bookmark tag has been reached. The Bookmark tag must be empty.

The Bookmark tag has the following attribute: Mark. Its value is a string. This value can be used to differentiate between bookmark events (each of which will contain the string value from their corresponding tag).

The application receives an event here:

```
<bookmark mark="bookmark_one"/>
```

and another event here:

```
<bookmark mark="bookmark_two"/>
```

Voice context control tags

Two tags provide context to the current voice: PartOfSp and Context. These tags enable the voice to determine how to deal with the text it is processing. With both of these tags, the extent to which voices use the context may vary.

PartOfSp

The PartOfSp tag provides the voice with the part of speech of the enclosed word(s). Use this tag to enable the voice to pronounce a word with multiple pronunciations correctly depending on its part of speech. The PartOfSp tag cannot be empty.

The PartOfSp tag has the following attribute: Part. It takes a string corresponding to a SAPI part of speech as its attribute. Only the followings SAPI defined parts of speech are supported: "Unknown", "Noun", "Verb", "Modifier", "Function", and "Interjection".

```
<partofsp part="noun"> A </partofsp> is the first letter of the
alphabet.
Did you <partofsp part="verb"> record </partofsp> that <partofsp
part="noun"> record </partofsp>?
```

Context

The Context tag provides the voice with information it uses to determine how to normalize special items (for e.g., dates, numbers, and currency). Use this tag to enable the voice to distinguish between confusable date formats (see the example, below). The Context tag cannot be empty.
The Context tag has one attribute, Id, which takes a string corresponding to the context of the enclosed text. Several contexts are defined by SAPI and are more likely to be recognized by SAPI compliant voices, but any string may be used.

```
<context id="date_mdy"> 03/04/01 </context> should be March
fourth, two thousand one.
<context id="date_dmy"> 03/04/01 </context> should be April
third, two thousand one.
<context id="date_ymd"> 03/04/01 </context> should be April
first, two thousand four.
```

Voice Selection Tags

There are two tags that can be used to change the current voice: Voice and Lang.

Voice

The Voice tag selects a voice based on its qualities: Age, Gender, Language, Name, Vendor, and VendorPreferred. If the tag is empty, it changes the voice for all subsequent text. If it has content, it only changes the voice for that content. The Voice tag has the following two attributes: Required and Optional. These correspond exactly to the required and optional attribute parameters of the ISpObjectTokenCategory_EnumerateTokens and SpFindBestTokenfunctions. The selected voice follows exactly the same rules as the latter of these two functions. All the required attributes are present. More optional attributes are present than with the other installed voices. If several voices have equal numbers of optional attributes, one is selected at random.

In addition, when the Voice tag is used, the attributes of the current voice are always added as optional attributes. Therefore, a voice that is more similar to the current voice will be selected over one that is less similar. If no voice is found that matches all of the required attributes, no voice change will occur.

The default voice should speak this sentence:

```
<voice required="Gender=Female;Age!=Child">
```

A female non-child should speak this sentence:

```
<voice required="Age=Teen"
```

A teen should speak this sentence - if a female, non-child teen is present, she will be selected over a male teen, for example.

```
</voice>
</voice>
```

Lang

The Lang tag selects a voice based solely on its Language attribute. If the tag is empty, it changes the voice for all subsequent text. If the tag has content, it only changes the voice for that content.
The Lang tag has the following attribute: LangId. This attribute should be a LANGID (for e.g., 409 for U.S. English or 411 for Japanese). **Note:** these numbers are hexadecimal, but without the typical "0x".

The Lang tag is a shortened version of the Voice tag with the required attribute containing "Language=xxx". So, the following examples produce exactly the same results:

```
<voice required="Language=409">
```

A U.S. English voice should speak this:

```
</voice>
<lang langid="409">
```

A U.S. English voice should speak this:

```
</lang>
```

Custom Pronunciation

An alternative to using the <P> tag with the DISP and PRON attributes is to use custom pronunciation.

Using custom pronunciation, tags in the form of the following:

```
<P DISP="disp" PRON="pron">word</P>
```

can be written as:

```
<P>/disp/word/pron;</P>
```

More specifically, if you want to recognize the word hello only when it is pronounced as ah and display a greeting when it is recognized, use the following:

```
<P DISP="greeting" PRON="ah">hello</P>
```

Using custom pronunciation, it translates as follows:

```
<P>/greeting/hello/ah;</P>
```

Social Media Monitoring

Social Media Monitoring allows Central Stations to monitor Twitter and Facebook for malicious, informational, and brand-related content. The signals and alarms that are generated and sent into Manitou can be ranked based on keyword searches you set up in the MediaGateway 2. These searches are configurable based on keywords, negative weighting, and Geo-Fence technology. Keywords can be rated positively and negatively to escalate certain search terms and downplay others. If for example, you are searching the word "gun" for potential risk, you can add the word "arm" as a negative keyword. So, if someone tweets "that guy has a gun for an arm", the score is offset by the negative keyword, downgrading a potentially hazardous posting.

Note: Posts and tweets are live and unfiltered. Therefore, they could potentially display explicit or inappropriate language.

Setting up Social Media Monitoring

Importing a Social Media Monitoring Menu

You must import a menu for Social Media Montioring into the MediaGateway 2. For instructions on importing a MediaGateway 2 menu item, refer to "Importing and Exporting Menus" document.

Adding a Line Driver

You must add a Social Media Monitoring Line Driver.

For instructions on adding a MediaGateway 2 Line Driver, refer to "Line Drivers" document. For instructions on using Parameters when setting up a Line Driver, refer to the "Parameters" document.

Note: if Social Media Monitoring is not available as a MediaGateway 2 Line Function, you will need to import it. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

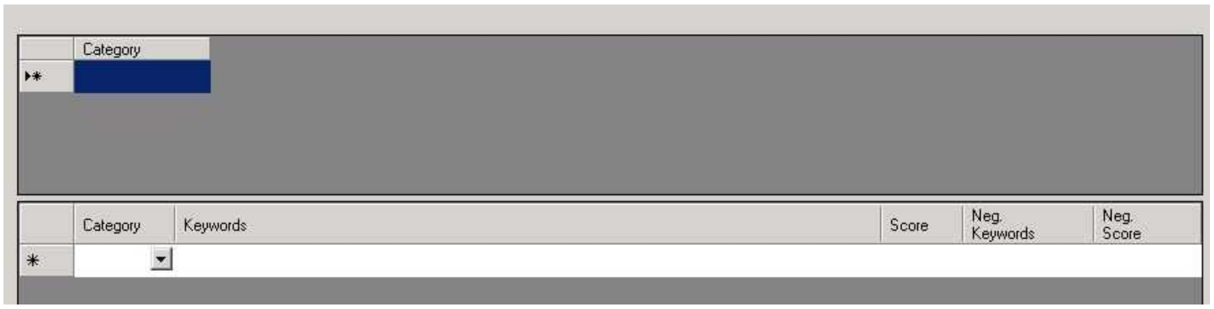
Social Media Searches

In order to search keywords with Social Media Monitoring, you must first establish and score the keywords you want to use.

Establishing and Scoring Keywords

Perform the following steps to establish and score keywords:

1. Select "Keywords" from the MediaGateway 2 Navigation Tree under Social Media.
Result: the Keywords form displays as shown in the following screenshot:



2. Click the next available field in the Category area of the form, and then enter a Category heading.
3. Click "**Save**".

Scoring Keywords in the Keywords Table

The Keywords table is divided into positive and negative keywords. Positive keywords add to a potential score and negative keywords subtract from it.

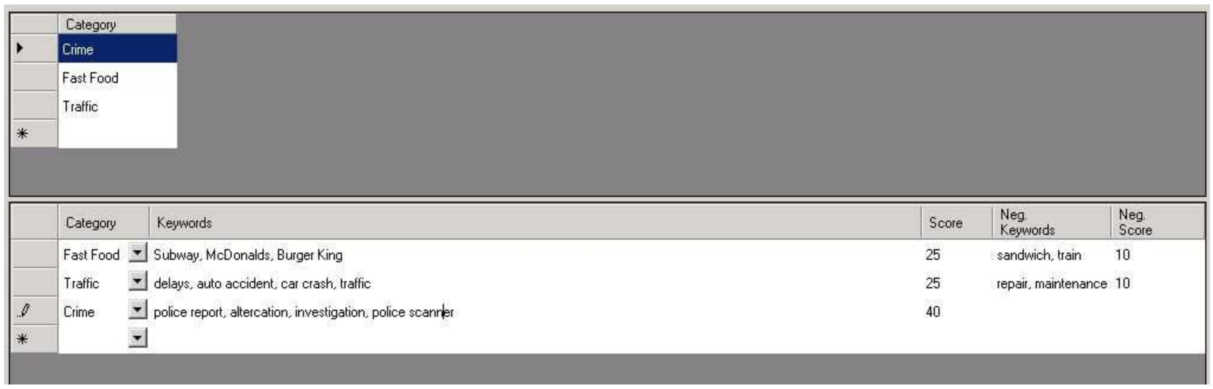
Perform the following steps to score keywords:

1. Select the new Category from the dropdown menu in the Keywords Table area of the window.
2. Enter positive keywords that pertain to the search in the "Keywords" column. Separate

each word with a comma.
3. Designate the score for each keyword in the "Score" column.

Note: scores range from 1 to 100.

4. Enter any negative keywords in the "Neg. Keywords" column. Separate each word with a comma.



	Category
▶	Crime
	Fast Food
	Traffic
*	

	Category	Keywords	Score	Neg. Keywords	Neg. Score
	Fast Food	Subway, McDonalds, Burger King	25	sandwich, train	10
	Traffic	delays, auto accident, car crash, traffic	25	repair, maintenance	10
✎	Crime	police report, altercation, investigation, police scanner	40		
*					

Categories and Keywords

5. Click in the "Category" field for the next line (indicated with an asterisk).

Note: this is necessary to ensure that the negative keywords you entered save properly.

6. Click "**Save**".
7. Repeat steps 1-6 for additional categories you want to score.

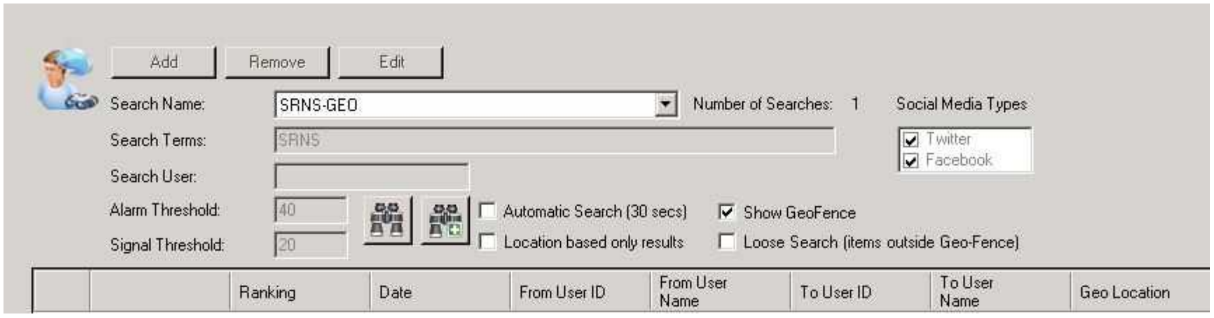
Defining a Social Media Search

After you have added and scored the keywords for your search, you must define search parameters.

Perform the following steps to define your search parameters:

1. Open the MediaGateway 2.
2. Click "Search" under Operations and Social Media in the Navigation Tree.

Result: the following window displays:

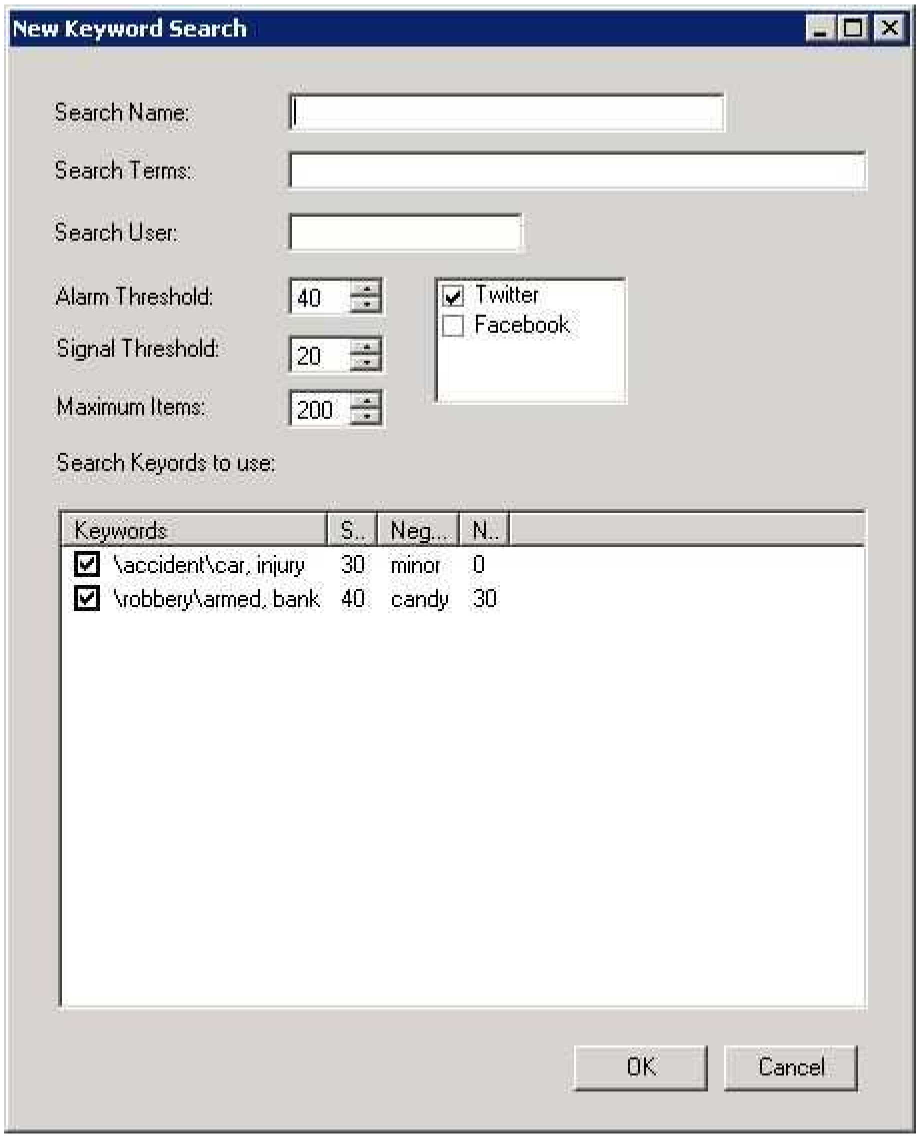


Search Name: SRNS-GEO
Number of Searches: 1
Social Media Types: ✓ Twitter ✓ Facebook
Search Terms: SRNS
Search User:
Alarm Threshold: 40
Signal Threshold: 20
☐ Automatic Search (30 secs)
☐ Location based only results
✓ Show GeoFence
☐ Loose Search (items outside Geo-Fence)

		Ranking	Date	From User ID	From User Name	To User ID	To User Name	Geo Location

3. Click "**Add**".

Result: the "New Keyword Search" window displays as shown in the following screenshot:



Search Name:

Search Terms:

Search User:

Alarm Threshold: 40

Signal Threshold: 20

Maximum Items: 200

☑ Twitter
☐ Facebook

Search Keyords to use:

	Keywords	S..	Neg...	N..
☑	\accident\car, injury	30	minor	0
☑	\robbery\armed, bank	40	candy	30

4. Enter a name for your new search in the "Search Name:" field.
5. Enter the search terms you want to use in the "Search Terms:" field. Separate each word with a comma.
6. Enter a user name in the "Search User:" field
7. If you want, modify the value in the the "Alarm Threshold:" field.
8. If you want, modify the value in the "Signal Threshold:" field.

Note: the Alarm Threshold is the minimum score with which an identified post will generate an alarm. The default value is 40. The Signal Threshold is the minimum score with which an identified post will generate a signal. The default value is set to 20.

9. If you want, modify the value in the "Maximum Items:" field.

Note: this value represents the maximum number of posts the MediaGateway 2 will identify for a single set of search terms. The default value is 200.

10. If you want to search Facebook as well as Twitter, select the Facebook checkbox.

11. If you want to limit the keywords the MediaGateway 2 uses to search, deselect the keyword sets you want to omit in the Keywords area of the window.
12. Click "**OK**".

Result: the "New Keyword Search" window closes and the system returns the user to the original search window.

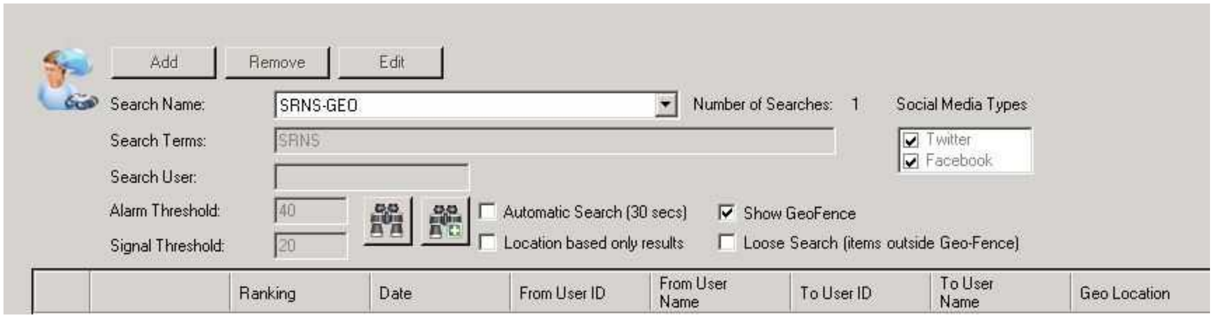
13. Click "**Save**".

Performing a Search in Social Media Monitoring

Perform the following steps to search for keywords using Social Media Monitoring:

1. Open the MediaGateway 2.
2. Click "Search" under Administration and Social Media in the Navigation Tree.

Result: the following window displays:

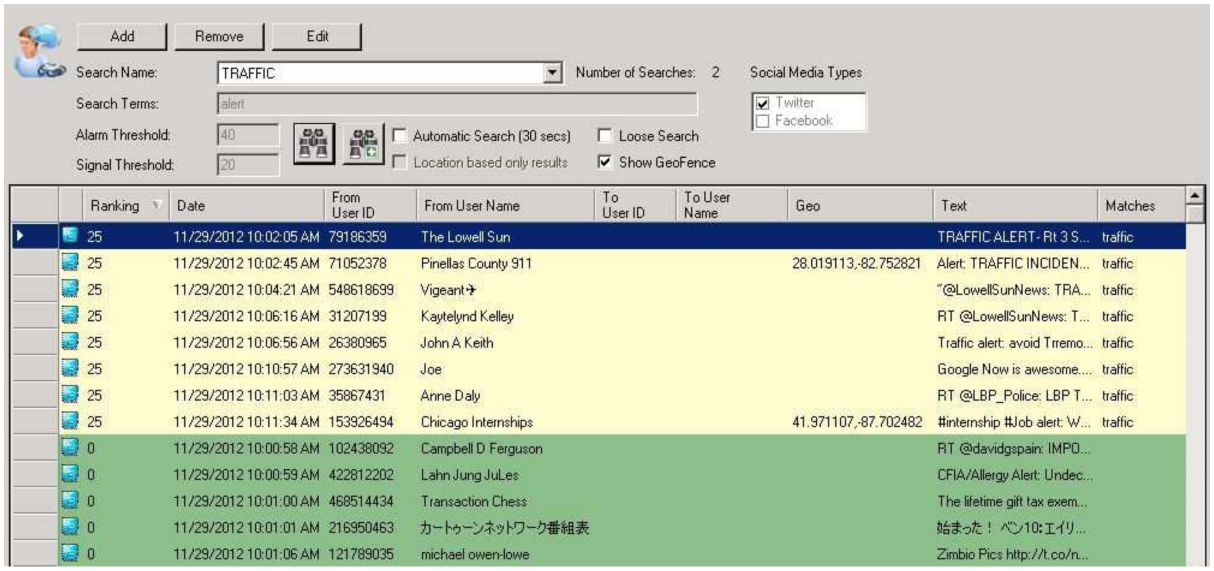


Search Name: SRNS-GEO
Number of Searches: 1
Social Media Types: [✓ Twitter] [✓ Facebook]
Search Terms: SRNS
Search User:
Alarm Threshold: 40
Signal Threshold: 20
☐ Automatic Search (30 secs)
☐ Location based only results
☑ Show GeoFence
☐ Loose Search (items outside Geo-Fence)

		Ranking	Date	From User ID	From User Name	To User ID	To User Name	Geo Location

3. Select the saved search you want from the "Search Name:" dropdown menu.
4. Click the "Search" icon as displayed in the following screenshot:

Result: your search displays as shown in the following screenshot:



Search Name: TRAFFIC
Number of Searches: 2
Social Media Types
[✓ Twitter] Facebook
Search Terms: alert
Alarm Threshold: 40
Signal Threshold: 20
Automatic Search (30 secs)
Location based only results
Loose Search
[✓ Show GeoFence]

Ranking	Date	From User ID	From User Name	To User ID	To User Name	Geo	Text	Matches
25	11/29/2012 10:02:05 AM	79186359	The Lowell Sun				TRAFFIC ALERT- Rt 3 S...	traffic
25	11/29/2012 10:02:45 AM	71052378	Pinellas County 911			28.019113,-82.752821	Alert: TRAFFIC INCIDEN...	traffic
25	11/29/2012 10:04:21 AM	548618699	Vigeant✈				"@LowellSunNews: TRA...	traffic
25	11/29/2012 10:06:16 AM	31207199	Kaytelynd Kelley				RT @LowellSunNews: T...	traffic
25	11/29/2012 10:06:56 AM	26380965	John A Keith				Traffic alert: avoid Trremo...	traffic
25	11/29/2012 10:10:57 AM	273631940	Joe				Google Now is awesome....	traffic
25	11/29/2012 10:11:03 AM	35867431	Anne Daly				RT @LBP_Police: LBP T...	traffic
25	11/29/2012 10:11:34 AM	153926494	Chicago Internships			41.971107,-87.702482	#internship #Job alert: W...	traffic
0	11/29/2012 10:00:58 AM	102438092	Campbell D Ferguson				RT @davidgspain: IMPO...	
0	11/29/2012 10:00:59 AM	422812202	Lahn Jung JuLes				CFIA/Allergy Alert: Undec...	
0	11/29/2012 10:01:00 AM	468514434	Transaction Chess				The lifetime gift tax exem...	
0	11/29/2012 10:01:01 AM	216950463	カートゥーンネットワーク番組表				始まった！ ベン10:エイリ...	
0	11/29/2012 10:01:06 AM	121789035	michael owen-lowe				Zimbio Pics http://t.co/n...	

Search Results

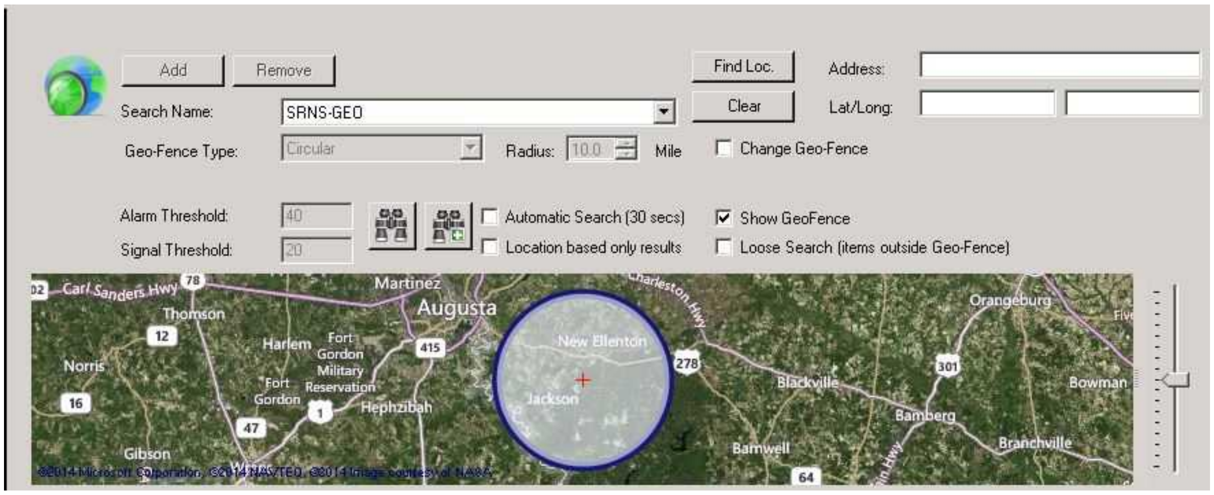
If you want to perform a subsequent search, click the "Add to Search" icon as displayed in the following screenshot:

Note: the "Add to Search" function maintains results from your original search, but also includes results for new social media messages posted after the original search completed.

Performing a Search with a Geo-Fence

1. Open the MediaGateway 2.
2. Click "Geo-Fence" from the Navigation Tree under Administration and Social Media.

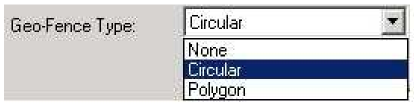
Result: the following window displays:



Address:
Lat/Long:
Search Name: SRNS-GEO
Geo-Fence Type: Circular
Radius: 10.0 Mile
Change Geo-Fence
Alarm Threshold: 40
Signal Threshold: 20
Automatic Search (30 secs)
Location based only results
[✓ Show GeoFence]
Loose Search (items outside Geo-Fence)

3. Select the saved search you want from the "Search Name:" field.
4. Select the "Change Geo-Fence" and "Show GeoFence" checkboxes.
5. Select an option from the "Geo-Fence Type:" dropdown menu.

Note: your options are "Circular" and "Polygon" as shown in the following screenshot:

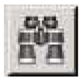


Note: Geo-Fences can extend up to a 100 mile radius from a central point.

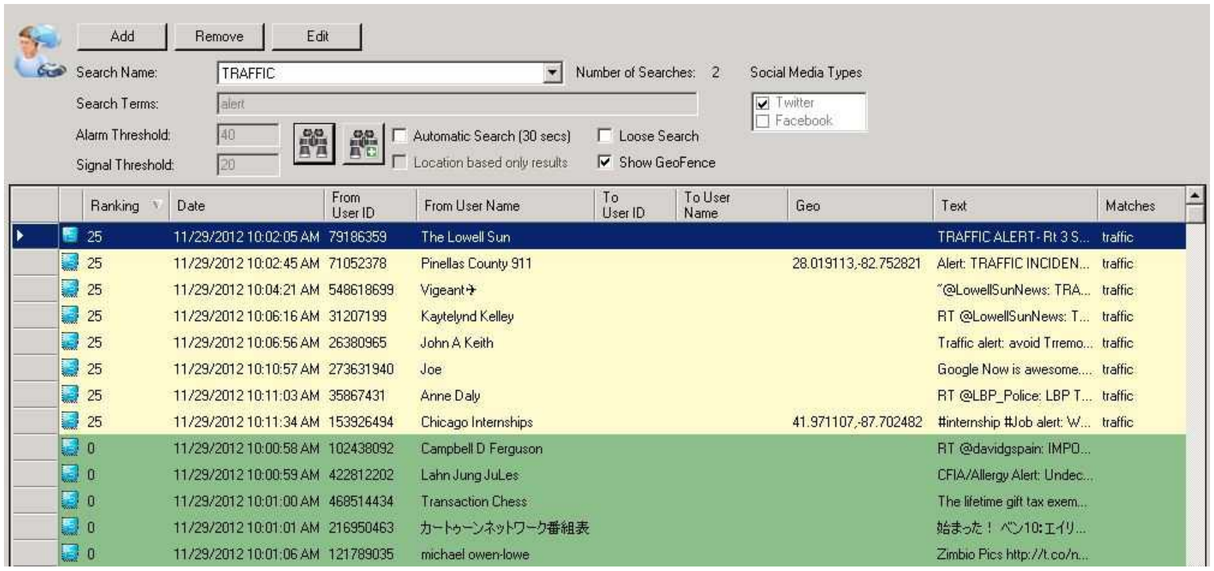
6. If you want, modify the value in the "Radius:" field. The default value is set to 10 miles.
7. If you want to modify the search area, right-click on the map central point (denoted by a red "X"), and drag it to the location you want.
8. If you want, select the "Loose Search" checkbox.

Note: selecting this item allows you to detect search items outside the defined Geo-Fence boundary.

9. Click the "Search" icon as displayed in the following screenshot:



Result: your search displays as shown in the following screenshot:



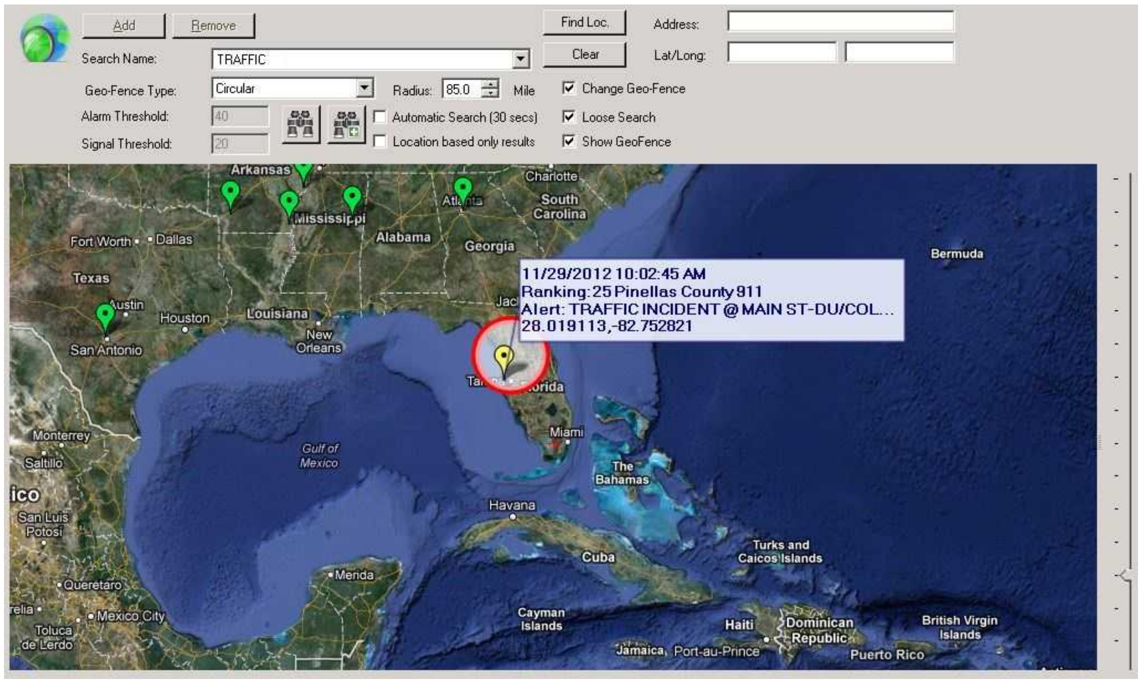
Search Name: TRAFFIC
Number of Searches: 2
Social Media Types: [✓ Twitter] Facebook
Search Terms: alert
Alarm Threshold: 40
Signal Threshold: 20
Automatic Search (30 secs)
Location based only results
Loose Search
[✓ Show GeoFence]

Ranking	Date	From User ID	From User Name	To User ID	To User Name	Geo	Text	Matches
25	11/29/2012 10:02:05 AM	79186359	The Lowell Sun				TRAFFIC ALERT - Rt 3 S...	traffic
25	11/29/2012 10:02:45 AM	71052378	Pinellas County 911			28.019113,-82.752821	Alert: TRAFFIC INCIDEN...	traffic
25	11/29/2012 10:04:21 AM	548618699	Vigeant✈				"@LowellSunNews: TRA...	traffic
25	11/29/2012 10:06:16 AM	31207199	Kaytelynd Kelley				RT @LowellSunNews: T...	traffic
25	11/29/2012 10:06:56 AM	26380965	John A Keith				Traffic alert: avoid Trremo...	traffic
25	11/29/2012 10:10:57 AM	273631940	Joe				Google Now is awesome....	traffic
25	11/29/2012 10:11:03 AM	35867431	Anne Daly				RT @LBP_Police: LBP T...	traffic
25	11/29/2012 10:11:34 AM	153926494	Chicago Internships			41.971107,-87.702482	#internship #Job alert: W...	traffic
0	11/29/2012 10:00:58 AM	102438092	Campbell D Ferguson				RT @davidgspain: IMPO...	
0	11/29/2012 10:00:59 AM	422812202	Lahn Jung JuLes				CFIA/Allergy Alert: Undec...	
0	11/29/2012 10:01:00 AM	468514434	Transaction Chess				The lifetime gift tax exem...	
0	11/29/2012 10:01:01 AM	216950463	カートゥーンネットワーク番組表				始まった！ ベン10:エイリ...	
0	11/29/2012 10:01:06 AM	121789035	michael owen-lowe				Zimbio Pics http://t.co/n...	

Search Results

10. Click a search result item to launch a new window with additional information about the post or tweet.
11. Geo-Fence results on the map display in the following manner:
    - Green tags - posts or tweets that do not score within the threshold limits
    - Yellow tags - posts or tweets that have generated signals

- Red tags - posts or tweets that have generated alarms



Search Name: TRAFFIC
Geo-Fence Type: Circular
Radius: 85.0 Mile
Alarm Threshold: 40
Signal Threshold: 20
Address:
Lat/Long:
☐ Automatic Search (30 secs)
☐ Location based only results
☑ Change Geo-Fence
☑ Loose Search
☑ Show GeoFence

11/29/2012 10:02:45 AM
Ranking:25 Pinellas County 911
Alert: TRAFFIC INCIDENT @ MAIN ST-DU/COL...
28.019113,-82.752821

Loose Geo-Fence search, Traffic results

SoloProtect

SoloProtect appears as a company badge holder, but is really a one-way cell phone device designed for personal protection. When the user presses the button, the device calls into the MediaGateway 2 and opens up a channel that records audio. A Manitou Operator then receives the recorded audio slices as voice clips.

Setting Up SoloProtect in the MediaGateway 2

Importing a SoloProtect Menu

You must import a SoloProtect menu into the MediaGateway 2. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Adding a Line Driver

You must add a SoloProtect Line Driver.

For instructions on adding a MediaGateway 2 Line Driver, refer to the "Line Drivers" document.
For instructions on using Parameters when setting up a Line Driver, refer to the "Parameters" document.

Note: if SoloProtect is not available as a MediaGateway 2 Line Function, you will need to import the menu. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

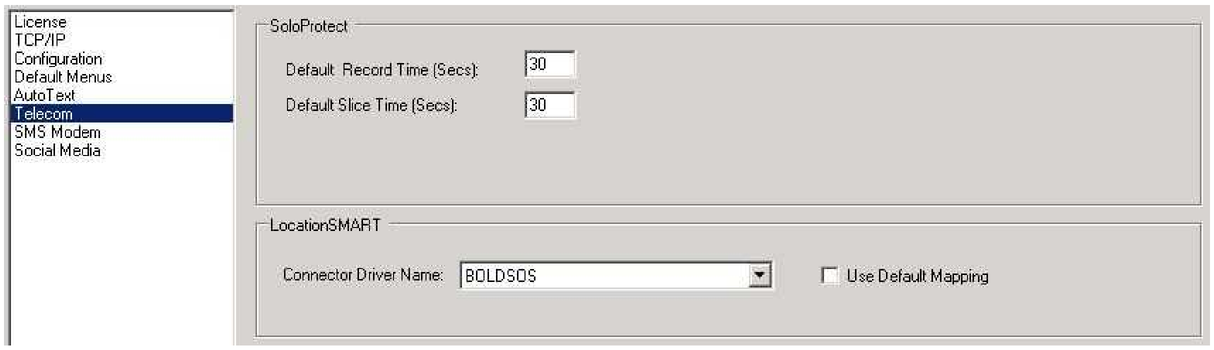
Designating a Default Menu

You must designate a default menu for SoloProtect to use if no other menus have been designated.

For instructions on designating default menus, refer to the "Default Menus" document.

Setting Up SoloProtect in the MediaGateway 2 Options Menu

The SoloProtect form in the MediaGateway 2 Options menu allows you to set the default times (in seconds) for recording and slicing audio.



Default Record Time (Secs): 30
Default Slice Time (Secs): 30
Connector Driver Name: BOLDSOS

Default Record Time: the total number of seconds you want to allow for voice recording. The default value is 30.

Default Slice Time: the total number of seconds for each slice of the total record time. The default value is 30.
(For e.g., if the record time is set to 30 seconds and the slice time is set to 10 seconds, then you will derive 3 10-second audio slices. The default values yield only a single audio slice).

Adding an SMS Connector

SoloProtect transmits information in two parts. The first transmission is an SMS message that sends the alarm into Manitou. The second transmission is a phone call that contains the audio slices detailed in the previous section.

If you have not already done so, you must set up the SMS Connector for the UniversalConnector. For instructions on setting up the SMS Connector, refer to the "SMS" document in the "Connector Setup" section.

Setting up SoloProtect in Manitou

You must add a GPS system to the Customer record in the Manitou Operator Workstation.

The Transmitter Type must match the SoloProtect Transmitter Type. The caller ID must match the Transmitter ID.

For more information on setting up a Customer record and adding a Transmitter, please refer to the Operator Workstation Manual for your version of Manitou.

TwoWayPLUS

TwoWayPLUS provides a fully integrated two-way voice solution. TwoWayPLUS allows call-follow technology in which two-way voice alarms and two-way voice calls are routed together to the correct Manitou Operator.

- TwoWayVoice – Keeps related two-way voice alarms and two-way voice calls together.
- InstantConnect – Turns virtually any phone into an alarm system.

TwoWayVoice

TwoWayVoice allows for a two-way voice call to be automatically transferred to the

Operator who is handling a related two-way voice alarm.

The following example highlights how TwoWayVoice helps a Manitou Operator efficiently handle an alarm:

A call arrives from the panel into the MediaGateway 2. The call is automatically connected to the Receiver. The Receiver communicates to Manitou that it can expect a two-way voice call from the user's caller ID. When the Receiver finishes with the call, the MediaGateway 2 awaits further instructions. The system sends "Keep-Alive" signals to the alarm panel to keep it online. Before the call is received by the Manitou Operator, Manitou communicates to the MediaGateway 2 the extension to which it must transfer the call. When the Receiver sends the call into Manitou, the call is transferred to the Operator the moment he picks up the alarm.

Setting Up TwoWayVoice in the MediaGateway 2

Importing a TwoWayVoice Menu

You must import a TwoWayVoice menu into the MediaGateway 2. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Adding a Line Driver

You must add a TwoWayVoice Line Driver.

For instructions on adding a MediaGateway 2 Line Driver, refer to the "Line Drivers" document.
For instructions on using Parameters when setting up a Line Driver, refer to the "Parameters" document.

Note: if TwoWayVoice is not available as a MediaGateway 2 Line Function, you will need to import the menu. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Designating a Default Menu

If you have not already done so, you must designate a default menu forTwoWayVoice to use if no other menus have been designated.

For instructions on designating default menus, refer to "Default Menus" document.

Setting up TwoWayVoice in Manitou

You must add an Event Monitoring System to the Customer record in the Manitou Operator Workstation.

The Transmitter Type must match the TwoWayVoice Transmitter Type. The caller ID must match the Transmitter ID.

For more information on setting up a Customer record and adding a Transmitter, please refer to the Operator Workstation Manual for your version of Manitou.

InstantConnect

The Bold InstantConnect module enables any phone to become an alarm system. When a telephone call comes into a Central Station, InstantConnect uses the caller ID to generate an alarm in Manitou. When a Manitou Operator picks up the alarm, the voice call is automatically routed to that Operator. InstantConnect uses caller ID in place of a Transmitter ID, so no Receiver is required.

Setting Up InstantConnect in the MediaGateway 2

Importing an InstantConnect Menu

You must import an InstantConnect menu into the MediaGateway 2. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Adding a Line Driver

You must add an InstantConnect Line Driver.

For instructions on adding a MediaGateway 2 Line Driver, refer to the "Line Drivers" document.
For instructions on using Parameters when setting up a Line Driver, refer to the "Parameters" document.

Note: if InstantConnect is not available as a MediaGateway 2 Line Function, you will need to import the menu. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Designating a Default Menu

You must designate a default menu for InstantConnect to use if no other menus have been designated.

For instructions on designating default menus, refer to "Default Menus" document.

Setting up InstantConnect in Manitou

You must add an Event Monitoring System to the Customer record in the Manitou Operator Workstation.

The Transmitter Type must match the InstantConnect Transmitter Type. The caller ID must match the Transmitter ID.

For more information on setting up a Customer record and adding a Transmitter, please refer to the Operator Workstation Manual for your version of Manitou.

Manitou Setup

You must set up an event monitoring system in the Operator Workstation to receive the alarm data.

The Transmitter Type must match the InstantConnect Transmitter Type and the Caller ID from the unit (used as the Transmitter ID).

For more information on setting up a Customer record and Transmitters, please refer to the appropriate user manual for your version of Manitou.

MediaGateway Setup

Once the initial configuration is finished in Manitou, you can move on to the MediaGateway to finish setting up InstantConnect.

Import an InstantConnect Menu

Begin by importing a menu for InstantConnect.

You must import an InstantConnect menu into the MediaGateway 2. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Adding a Line Driver

You must add an InstantConnect Line Driver.

For instructions on adding a MediaGateway 2 Line Driver, refer to the "Line Drivers" document.
For instructions on using Parameters when setting up a Line Driver, refer to the "Parameters" document.

Note: if InstantProtect is not available as a MediaGateway 2 Line Function, you will need to import the menu. For instructions on importing a MediaGateway 2 menu item, refer to the "Importing and Exporting Menus" document.

Designating a Default Menu

You must designate a default menu for InstantConnect to use if no other menus have been designated.

For instructions on designating default menus, refer to the "Default Menus" document.

Troubleshooting

The Bold Knowledgebase is available as a reference for issues related to the MediaGateway 2.

Accessing the Knowledgebase

Perform the following steps to access the Knowledgebase

1. Log on to https://support.boldgroup.com
2. Enter the username and password provided to you by Bold. If you do not yet have a username and password for the Knowledgebase, contact Bold.
3. View the Knowledgebase and access the MediaGateway 2 category.

Note: if your particular issue has not yet been documented in the Knowledgebase, open a ticket to get the issue reviewed by a member of the Bold staff.

Appendix A - M.F.C.s Full List

A full list of M.F.C.s has been provided in this section for your convenience.

Note: these M.F.C.s are not formatted, and have been provided in their original script.

General M.F.C.s

Add("**ClearCache**", clsGlobals.s_Language.GetText("Clear Data Cache"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

Add("**ClearInputBuffer**", clsGlobals.s_Language.GetText("Clear Input Buffer"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enUnassigned);

Add("**EndSession**", clsGlobals.s_Language.GetText("End Session"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Message", null, clsGlobals.s_Language.GetText("Session Ended"), enVariantType.String, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetBooleanParamInput**", clsGlobals.s_Language.GetText("Get Single Execute Parameter Input"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("BooleanParameter1", false, "{$TBOOL1#}", enVariantType.Boolean, clsGlobals.s_Language.GetText("Enter bool Parameter 1"), "", false, clsSvrFunctionParmameter.enModifier.enStore);

Add("**GetMultiParamInput**", clsGlobals.s_Language.GetText("Get Multi-Parameter Input"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

```
AddParameters("StringParameter1", "", "{$TSTR1#}", enVariantType.String,
clsGlobals.s_Language.GetText("Enter string Parameter 1"), "", false,
clsSvrFunctionParmameter.enModifier.enStore);

AddParameters("BooleanParameter1", false, "{$TBOOL1#}", enVariantType.Boolean,
clsGlobals.s_Language.GetText("Enter bool Parameter 1"), "", false,
clsSvrFunctionParmameter.enModifier.enStore);

AddParameters("StringParameter2", "", "{$TSTR2#}", enVariantType.String,
clsGlobals.s_Language.GetText("Enter string Parameter 2"), "", false,
clsSvrFunctionParmameter.enModifier.enStore);

AddParameters("BooleanParameter2", false, "{$TBOOL2#}", enVariantType.Boolean,
clsGlobals.s_Language.GetText("Enter bool Parameter 2"), "", false,
clsSvrFunctionParmameter.enModifier.enStore);

AddParameters("StringParameter3", "", "{$TSTR3#}", enVariantType.String,
clsGlobals.s_Language.GetText("Enter string Parameter 3"), "", false,
clsSvrFunctionParmameter.enModifier.enStore);

AddParameters("BooleanParameter3", false, "{$TBOOL3#}", enVariantType.Boolean,
clsGlobals.s_Language.GetText("Enter bool Parameter 3"), "", false,
clsSvrFunctionParmameter.enModifier.enStore);

AddParameters("StringParameter4", "", "{$TSTR4#}", enVariantType.String,
clsGlobals.s_Language.GetText("Enter string Parameter 4"), "", false,
clsSvrFunctionParmameter.enModifier.enStore);

AddParameters("BooleanParameter4", false, "{$TBOOL4#}", enVariantType.Boolean,
clsGlobals.s_Language.GetText("Enter bool Parameter 4"), "", false,
clsSvrFunctionParmameter.enModifier.enStore);

AddParameters("StringParameter5", "", "{$TSTR5#}", enVariantType.String,
clsGlobals.s_Language.GetText("Enter string Parameter 5"), "", false,
clsSvrFunctionParmameter.enModifier.enStore);

AddParameters("BooleanParameter5", false, "{$TBOOL5#}", enVariantType.Boolean,
clsGlobals.s_Language.GetText("Enter bool Parameter 5"), "", false,
clsSvrFunctionParmameter.enModifier.enStore);


Add("GetRow", clsGlobals.s_Language.GetText("Get Row"), Execute,
Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Table", null, "", enVariantType.String, "Right Click to Choose Table",
"TABLE", false, clsSvrFunctionParmameter.enModifier.enNothing);
```

AddParameters("Direction", null, "CURR", enVariantType.String, "Right Click to Choose Direction", "DIRECTION", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetStringParamInput**", clsGlobals.s_Language.GetText("Get Single string Parameter Input"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("StringParameter1", "", "{$TSTR1#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter string Parameter 1"), "", false, clsSvrFunctionParmameter.enModifier.enStore);

Add("**HasKeyBeenPressed**", clsGlobals.s_Language.GetText("Has 'key' been pressed"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Key", null, "1", enVariantType.String, clsGlobals.s_Language.GetText("Enter Key Value"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**IfCounterEqual**", clsGlobals.s_Language.GetText("If Counter Equal"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Counter", null, 1, enVariantType.Integer, clsGlobals.s_Language.GetText("Enter Counter Number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("TestValue", null, null, enVariantType.Integer, clsGlobals.s_Language.GetText("Enter Test Value"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Global", null, 0, enVariantType.Boolean, clsGlobals.s_Language.GetText("Enter Value True for Global Counter"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**IfCounterGreater**", clsGlobals.s_Language.GetText("If Counter Greater"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Counter", null, 1, enVariantType.Integer, clsGlobals.s_Language.GetText("Enter Counter Number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("TestValue", null, null, enVariantType.Integer, clsGlobals.s_Language.GetText("Enter Test Value"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Global", null, 0, enVariantType.Boolean, clsGlobals.s_Language.GetText("Enter Value True for Global Counter"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add(""**IfCounterGreaterEqual**"", clsGlobals.s_Language.GetText(""If Counter Greater or Equal""), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);"

AddParameters(""Counter"", null, 1, enVariantType.Integer, clsGlobals.s_Language.GetText(""Enter Counter Number""), """", false, clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""TestValue"", null, null, enVariantType.Integer, clsGlobals.s_Language.GetText(""Enter Test Value""), """", false, clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""Global"", null, 0, enVariantType.Boolean, clsGlobals.s_Language.GetText(""Enter Value True for Global Counter""), """", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add(""**IfCounterLess**"", clsGlobals.s_Language.GetText(""If Counter Less""), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);"

AddParameters(""Counter"", null, 1, enVariantType.Integer, clsGlobals.s_Language.GetText(""Enter Counter Number""), """", false, clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""TestValue"", null, null, enVariantType.Integer, clsGlobals.s_Language.GetText(""Enter Test Value""), """", false, clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""Global"", null, 0, enVariantType.Boolean, clsGlobals.s_Language.GetText(""Enter Value True for Global Counter""), """", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add(""**IfCounterLessEqual**"", clsGlobals.s_Language.GetText(""If Counter Less or Equal""), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);"

AddParameters(""Counter"", null, 1, enVariantType.Integer, clsGlobals.s_Language.GetText(""Enter Counter Number""), """", false, clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""TestValue"", null, null, enVariantType.Integer, clsGlobals.s_Language.GetText(""Enter Test Value""), """", false, clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""Global"", null, 0, enVariantType.Boolean, clsGlobals.s_Language.GetText(""Enter Value True for Global Counter""), """", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**IfStringEmpty**", clsGlobals.s_Language.GetText("IF string is Empty then Success"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("StringParameter1", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter string Parameter 1"), "", false, clsSvrFunctionParmameter.enModifier.enIgnore);

Add("**IfTrue**", clsGlobals.s_Language.GetText("If True then Success"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("BooleanTest", null, null, enVariantType.Boolean, clsGlobals.s_Language.GetText("Enter bool Test"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**IfStringInList**", clsGlobals.s_Language.GetText("IF String1 is in comma separated String2 then Success"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("StringParameter1", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter string Parameter 1"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("StringParameter2", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter comma separated strings Parameter 2"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("IgnoreCase", null, true, enVariantType.Boolean, clsGlobals.s_Language.GetText("Enter True to Ignore Case"), "BOOLEAN", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**IncrementCounter**", clsGlobals.s_Language.GetText("Increment Counter"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Counter", null, 1, enVariantType.Integer, clsGlobals.s_Language.GetText("Enter Counter Number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Global", null, 0, enVariantType.Boolean, clsGlobals.s_Language.GetText("Enter Value True for Global Counter"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**IsFirstRow**", clsGlobals.s_Language.GetText("Is First Row"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Table", null, "", enVariantType.String, "Right Click to Choose Table", "TABLE", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**IsLastRow**", clsGlobals.s_Language.GetText("Is Last Row"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Table", null, "", enVariantType.String, "Right Click to Choose Table", "TABLE", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**IsTCPConnected**", clsGlobals.s_Language.GetText("TCP - Is Connected"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

Add("**None**", clsGlobals.s_Language.GetText("Nothing"), (clsSvrFunctions.SvrFunction)this.Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

Add("**SetVariableToSpeak**", clsGlobals.s_Language.GetText("Set variable into {$SPEAK#}"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Speak", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter string to be converted to speak"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

```
Add("Sleep", clsGlobals.s_Language.GetText("Sleep"), Execute, Constants.MenuType_Any,
clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Milliseconds", null, "1000", enVariantType.Integer,
clsGlobals.s_Language.GetText("Enter Milliseconds to sleep"), "", false,
clsSvrFunctionParmameter.enModifier.enNothing);

 Add("SetCounter", clsGlobals.s_Language.GetText("Set Counter"), Execute,
Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Counter", null, 1, enVariantType.Integer,
clsGlobals.s_Language.GetText("Enter Counter Number"), "", false,
clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Value", null, 0, enVariantType.Integer,
clsGlobals.s_Language.GetText("Enter Value to Set"), "", false,
clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Global", null, 0, enVariantType.Boolean,
clsGlobals.s_Language.GetText("Enter Value True for Global Counter"), "", false,
clsSvrFunctionParmameter.enModifier.enNothing);


Add("TCPConnect", clsGlobals.s_Language.GetText("TCP - Open a Connection"), Execute,
Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Server", null, null, enVariantType.String,
clsGlobals.s_Language.GetText("Server to connect to"), "", false,
clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Port", null, null, enVariantType.String,
clsGlobals.s_Language.GetText("Port to connect to"), "", false,
clsSvrFunctionParmameter.enModifier.enNothing);


Add("TCPDisonnect", clsGlobals.s_Language.GetText("TCP - Disconnect a Connection"),
Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

Add("PlayWaveFile", clsGlobals.s_Language.GetText("Play a wave file"), Execute,
Constants.MenuType_Any, clsSvrFunctions.enCategory.enPhone);

AddParameters("File", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter
file name to play"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);
```

Add("**TCPRead**", clsGlobals.s_Language.GetText("TCP - Read data from a Connection"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

Add("**TCPWrite**", clsGlobals.s_Language.GetText("TCP - Write data to a Connection"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Output", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Output data"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**TTSConfiguration**", clsGlobals.s_Language.GetText("Set the TTS configuration"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("ConfigNo", null, 1, enVariantType.Integer, clsGlobals.s_Language.GetText("Enter the TTS Configuration number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**YesNo**", clsGlobals.s_Language.GetText("Yes/No"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enUnassigned);

AddParameters("Yes", null, "1", enVariantType.String, clsGlobals.s_Language.GetText("Enter Yes Value"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("No", null, "2", enVariantType.String, clsGlobals.s_Language.GetText("Enter No Value"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Dealer M.F.C.s

Add("**ClearDealerCache**", clsGlobals.s_Language.GetText("Clear Dealer Data Cache"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enDealer);

Add("**GetDealerByPhoneNumber**", clsGlobals.s_Language.GetText("Get Dealer by Phone Number"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enDealer);

AddParameters("Phone", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Phone Number"), "", false,

clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetDealerByContID**", clsGlobals.s_Language.GetText("Get Dealer by Contract ID"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enDealer);

AddParameters("ContID", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter dealer ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**LoadDealerbyDNIS**", clsGlobals.s_Language.GetText("Load Dealer by DNIS"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enDealer);

AddParameters("DNIS", null, "{$DNIS#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter DNIS"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("DNISPrefix", null, "{$DPRF#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter DNIS Prefix"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Customer M.F.C.s

Add("**ClearCustomerCache**", clsGlobals.s_Language.GetText("Clear Customer Data Cache"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

Add("**CommissionCustomer**", clsGlobals.s_Language.GetText("Change the Commission State of the Customer"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("CommState", null, 2, enVariantType.Integer, "Enter Commission State", "COMSTATE", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("DeleteCode", null, 0, enVariantType.Integer, "Enter Delete Code", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("DeleteReason", null, "", enVariantType.String, "Enter Delete Reason", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetCustomerArea**", clsGlobals.s_Language.GetText("Get Customer Area"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Area", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter area"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetCustomerByContID**", clsGlobals.s_Language.GetText("Get Customer by Contract ID"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enCustomer);

AddParameters("ContID", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter Contract ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add(""**GetCustomerByPhoneNumber**"", clsGlobals.s_Language.GetText(""Get Customer by Phone Number""), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);"

AddParameters(""Phone"", null, ""{$CID#}"", enVariantType.String, clsGlobals.s_Language.GetText(""Enter Phone Number""), """", false, clsSvrFunctionParmameter.enModifier.enNothing);"

Add("**GetCustomerByRLPrefixTxId**", clsGlobals.s_Language.GetText("Get Customer by Receiver Line Prefix and Transmitter ID"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enCustomer);

AddParameters("RLPrefix", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter receiver line prefix"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("TxID", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter transmitter ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetCustomerBySerialNo**", clsGlobals.s_Language.GetText("Get Customer by Serial Number"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetCustomerContactByPhoneNumber**", clsGlobals.s_Language.GetText("Get Customer Contact by Phone Number"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("Phone", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Phone Number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetCustomerForSelectedTicket**", clsGlobals.s_Language.GetText("Get Customer for Selected Ticket"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

Add("**GetCustomerOnTestParameters**", clsGlobals.s_Language.GetText("Get Customer On-Test Parameters"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("SystemNo", null, "{$SY#}", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter System Number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("CommsNo", null, "{$CN#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter comms number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Area", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter area"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Zone", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter zone"), "", false,

clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Minutes", null, null, enVariantType.Integer, clsGlobals.s_Language.GetText("Enter the number of minutes to put system on-test"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Related", null, false, enVariantType.Boolean, clsGlobals.s_Language.GetText("Enter True for related accounts"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetCustomerOnTestStatus**", clsGlobals.s_Language.GetText("Get Customer On-Test Status"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetCustomerZone**", clsGlobals.s_Language.GetText("Get Customer Zone"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Zone", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter zone"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetLastCustomer**", clsGlobals.s_Language.GetText("Get Last Customer for User"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("Hours", null, "8.0", enVariantType.Double, clsGlobals.s_Language.GetText("Enter the number of hours ago to get last customer"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetSingleCustomerAccess**", clsGlobals.s_Language.GetText("Get Customer if single access"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

Add("**GetTicket**", clsGlobals.s_Language.GetText("Get a Specific Ticket for a Technican"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("TechID", null, "{$TID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Technician ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("TicketNo", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter Technician ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Reserved", null, false, enVariantType.String, clsGlobals.s_Language.GetText("Reserved Events only"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**MarkSignalListenedTo**", clsGlobals.s_Language.GetText("Mark On-Test Signal as Listened to"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**MarkSignalXmittedTo**", clsGlobals.s_Language.GetText("Mark On-Test Signal as Transmitted to Technician"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**PutCustomerFullyOnTest**", clsGlobals.s_Language.GetText("Put Customer Fully On-Test"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Minutes", null, "{$MN#}", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter minutes to put on-test"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Related", null, "{$REL#}", enVariantType.Boolean, clsGlobals.s_Language.GetText("Enter True for related accounts"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**PutCustomerPartiallyOnTest**", clsGlobals.s_Language.GetText("Put Customer Partially On-Test"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("SystemNo", null, "{$SY#}", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter system Number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("CommsNo", null, "{$CN#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter comms number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Area", null, "{$AR#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter area"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Zone", null, "{$ZN#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter zone"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Minutes", null, "{$MN#}", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter minutes to put on-test"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Related", null, "{$REL#}", enVariantType.Boolean, clsGlobals.s_Language.GetText("Enter True for related accounts"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**RegisterForSMSNotify**", clsGlobals.s_Language.GetText("Register for SMS Notification"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("CallerID", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Caller ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SelectCustomerArea**", clsGlobals.s_Language.GetText("Select Customer Area"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SelectCustomerByPhoneNumber**", clsGlobals.s_Language.GetText("Select Customer by Phone Number"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("Phone", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Phone Number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SelectCustomerLog**", clsGlobals.s_Language.GetText("Select the Customer Log"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("CommsNo", null, "{$CN#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter comms number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("SystemNo", null, "{$SY#}", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter system Number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Minutes", null, null, enVariantType.Integer, clsGlobals.s_Language.GetText("Enter minutes ago for selection"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("EventCodeList", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter event code list"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("EventCatList", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter event category list"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SelectCustomerOnTestDetailsRows**", clsGlobals.s_Language.GetText("Select On-Test Details Rows"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SelectCustomerZones**", clsGlobals.s_Language.GetText("Select Customer Zones"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SelectOnTestSignals**", clsGlobals.s_Language.GetText("Select On-Test Signals Received"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SelectTicketList**", clsGlobals.s_Language.GetText("Select Service Tickets for a Technican"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("TechID", null, "{$TID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Technician ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Reserved", null, false, enVariantType.String, clsGlobals.s_Language.GetText("Reserved Events only"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SelectTransmitterList**", clsGlobals.s_Language.GetText("Select a List of All Transmitters"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Counter", null, 1, enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**StoreLastCustomer**", clsGlobals.s_Language.GetText("Store Last Customer for User"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**TakeOffTest**", clsGlobals.s_Language.GetText("Take the Current Row Off Test"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**UnregisterForSMSNotify**", clsGlobals.s_Language.GetText("Unregister for SMS Notification"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enCustomer);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("CallerID", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Caller ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Alarm M.F.C.s

Add("**AddAlarmComment**", clsGlobals.s_Language.GetText("Load Dealer by DNIS"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enDealer);

AddParameters("Text", null, null, enVariantType.String, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("TxId", null, null, enVariantType.String, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("SystemComment", null, true, enVariantType.Boolean, "Enter True for System Comment", "BOOLEAN", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**CancelAllAllocatedAlarms**", clsGlobals.s_Language.GetText("Cancel all Allocated Alarms"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enAlarm);

AddParameters("CancelOnly", null, false, enVariantType.Boolean, "Enter True to just Cancel, False to Close", "BOOLEAN", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**CancelCloseAlarm**", clsGlobals.s_Language.GetText("Cancel/Close Alarm"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enAlarm);

AddParameters("AlarmNo", null, "{$ANO#}", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter Alarm Number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("CancelOnly", null, false, enVariantType.Boolean, "Enter True to just Cancel, False to Close", "BOOLEAN", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Resolution", null, "", enVariantType.String, "Enter the resolution code for alarm", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**DidCustomerDisconnnect**", clsGlobals.s_Language.GetText("Did Customer Disconnect"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enAlarm);

Add("**EnterAlarmHandling**", clsGlobals.s_Language.GetText("Enter Alarm Handling"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enAlarm);

AddParameters("Locale", null, "USENG", enVariantType.String, clsGlobals.s_Language.GetText("Enter Manitou Locale"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**ExitAlarmHandling**", clsGlobals.s_Language.GetText("Exit Alarm Handling"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enAlarm);

Add("**ExtendCloseTime**", clsGlobals.s_Language.GetText("Extend Close Time"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enAlarm);

AddParameters("SerialNo", null, "{$ASN#}", enVariantType.Integer, "", "", false,

clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Area", null, "{$AAR#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter area"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Minutes", null, "", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter minutes to extend close time"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**ExtendOpenTime**", clsGlobals.s_Language.GetText("Extend Open Time"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enAlarm);

AddParameters("SerialNo", null, "{$ASN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Area", null, "{$AAR#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter area"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Minutes", null, "", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter minutes to extend open time"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetAlarm**", clsGlobals.s_Language.GetText("Get an Alarm"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enAlarm);

AddParameters("AlarmNo", null, "{$ANO#}", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter Alarm Number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**GetCustomerForAlarm**", clsGlobals.s_Language.GetText("Get Customer for Current Alarm (Out)"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enAlarm);

AddParameters("SerialNo", null, "{$ASN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**PreCancelAlarm**", clsGlobals.s_Language.GetText("Pre-Cancel an Alarm"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enAlarm);

AddParameters("SerialNo", null, "{$SN#}", enVariantType.Integer, "", "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Callback", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter Callback number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Comment", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter Comment"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

m_cFunc.m_cSvrFunctionParameters.AddParameters("Locale", null, "USENG", enVariantType.String, clsGlobals.s_Language.GetText("Enter Manitou Locale"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

cDataRow = m_cManitou.m_cPreCancel.Rows.Add();

Add("**RecordVoiceMessageAlarm**", clsGlobals.s_Language.GetText("Record Voice Message for Alarm"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enAlarm);

AddParameters("NoSecs", null, 60, enVariantType.Integer, clsGlobals.s_Language.GetText("Enter Number of Seconds to Record"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Tone", null, 1, enVariantType.Integer, clsGlobals.s_Language.GetText("Enter 1 for record Tone, 0 for no tone"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Comment", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter Comment Text"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SelectAlarmsByCallerID**", clsGlobals.s_Language.GetText("Select & Allocate Alarms by Caller ID"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enAlarm);

AddParameters("CallerID", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Caller ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Locale", null, "USENG", enVariantType.String, clsGlobals.s_Language.GetText("Enter Manitou Locale"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("LowPriority", null, "3", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter the lowest priority alarm allowed"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("DefOprExt", null, "0", enVariantType.String, clsGlobals.s_Language.GetText("Enter the default operator extension"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SendInvalidMessage**", clsGlobals.s_Language.GetText("Send an Invalid Message"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enAlarm);

AddParameters("Message", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter Message"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SendReceiverContactID**", clsGlobals.s_Language.GetText("Send a Receiver Contact ID Signal"), Execute, Constants.MenuType_LineCheck + Constants.MenuType_ChannelBank, clsSvrFunctions.enCategory.enAlarm);

AddParameters("Number", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter Number to dial"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Prefix", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter Dialing Prefix"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("TxID", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter transmitter ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Code", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter Contact ID Code"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Area", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter Area"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Zone", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter Zone"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SendSignal**", clsGlobals.s_Language.GetText("Send a Signal"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enAlarm);

AddParameters("TxID", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter transmitter ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("EvType", null, "SYS", enVariantType.String, clsGlobals.s_Language.GetText("Enter event type"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("StCode", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter standard event code"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Area", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter area"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Zone", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter zone"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Line", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter line number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("PointID", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter point ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("TwoWay", null, false, enVariantType.Boolean, clsGlobals.s_Language.GetText("Is Two-Way signal"), "BOOLEAN", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("CallerID", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Caller ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SendSignalWithAudio**", clsGlobals.s_Language.GetText("Send a Signal with Audio"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enAlarm);

AddParameters("TxID", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter transmitter ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("EvType", null, "SYS", enVariantType.String, clsGlobals.s_Language.GetText("Enter event type"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("StCode", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter standard event code"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Area", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter area"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Zone", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter zone"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("PointID", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter point ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Line", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter line number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("RecordTime", null, "60", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter the maximium time in seconds to record"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AutoText/UC M.F.C.s

Add("**SendSMS**", clsGlobals.s_Language.GetText("Send SMS Message"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enSMS);

AddParameters("Number", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter Phone Number to Send Message"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Message", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter Message"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Simulator", null, false, enVariantType.Boolean, clsGlobals.s_Language.GetText("Enter True for Simulated SMS"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add(""**UCDeleteNextSignal**"", clsGlobals.s_Language.GetText(""Delete next signal, do not send to Manitou""), Execute, Constants.MenuType_UniversalConnector, clsSvrFunctions.enCategory.enSMS);"

Add(""**UCIsNextSignal**"", clsGlobals.s_Language.GetText(""Is there a next signal""), Execute, Constants.MenuType_UniversalConnector, clsSvrFunctions.enCategory.enSMS);"

Add(""**UCModifyEvent**"", clsGlobals.s_Language.GetText(""Modify an Event code for the next prepared signal""), Execute, Constants.MenuType_UniversalConnector , clsSvrFunctions.enCategory.enSMS);"

AddParameters("StCodeFrom", null, "*", enVariantType.String, clsGlobals.s_Language.GetText("Enter standard event code to be changed from. '*' = any"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("StCodeTo", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter standard event code to be changed to"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add(""**UCModifyZone**"", clsGlobals.s_Language.GetText(""Modify a Zone for the next prepared signal.'*' = any""), Execute, Constants.MenuType_UniversalConnector, clsSvrFunctions.enCategory.enSMS);"

AddParameters("ZoneFrom", null, "*", enVariantType.String, clsGlobals.s_Language.GetText("Enter zone to be changed from"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("ZoneTo", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter zone to be changed to"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**UCPrepareSignal**", clsGlobals.s_Language.GetText("Prepare signals using mapped fields"), Execute, Constants.MenuType_UniversalConnector , clsSvrFunctions.enCategory.enSMS);

AddParameters("FieldSet", null, "{$FSET#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Field Set"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Signal", null, "{$SMS#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Signal"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("U**CSendPreparedSignal**", clsGlobals.s_Language.GetText("Send the next prepared signal using Mapped Fields"), Execute, Constants.MenuType_UniversalConnector | Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enSMS);

Add("**UCSendSignal**", clsGlobals.s_Language.GetText("Send and prepare all signals using Mapped Fields"), Execute, Constants.MenuType_UniversalConnector, clsSvrFunctions.enCategory.enSMS);

AddParameters("FieldSet", null, "{$FSET#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Field Set"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Signal", null, "{$SMS#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Signal"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Telecom M.F.C.s

Add("**BusyPort**", clsGlobals.s_Language.GetText("Make Line Port Ring Busy"), Execute, Constants.MenuType_ChannelBank, clsSvrFunctions.enCategory.enPhone);

AddParameters("Busy", null, true, enVariantType.Boolean, clsGlobals.s_Language.GetText("Enter True or False to Make Port Busy/Normal"));

Add("**FlushSMSBuffer**", clsGlobals.s_Language.GetText("Flush (Send) SMS Text Buffer"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enPhone);

Add("**Hangup**", clsGlobals.s_Language.GetText("Hangup the call"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enPhone);

Add("**IsCallInProgress**", clsGlobals.s_Language.GetText("Is Call in Progress"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enPhone);

Add("**SetCallResult**", clsGlobals.s_Language.GetText("Set the result of the call"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enPhone);

AddParameters("Result", null, false, enVariantType.Boolean, clsGlobals.s_Language.GetText("Enter Call Result"), "BOOLEAN", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Comment", "", "", enVariantType.String, clsGlobals.s_Language.GetText("Enter Result Comment"));

Add("**TransferCall**", clsGlobals.s_Language.GetText("Transfer OpenVoice/TwoWay call to number"), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enPhone);

AddParameters("Number", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter Phone Number to Transfer to"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Index", null, "1", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter the Phone Number Index"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**WaitForDisconnectFromTransfer**", clsGlobals.s_Language.GetText("Wait for Operator to disconnect from Transfer"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enPhone);

TwoWayPLUS M.F.C.s

Add("**AgentRegisterCallSession**", clsGlobals.s_Language.GetText("Register Agent Call with Manitou"), Execute, Constants.MenuType_TwoWay, clsSvrFunctions.enCategory.enTwoWay);

AddParameters("CallKey", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter key for the call"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**SoloProtectRegisterCallSession**", clsGlobals.s_Language.GetText("Register SoloProtect

Call with Manitou"), Execute, Constants.MenuType_TwoWay, clsSvrFunctions.enCategory.enTwoWay);

AddParameters("CallKey", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter key for the call"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**TwoWayAdoptCallSession**", clsGlobals.s_Language.GetText("Adopt an Existing Two-Way Call with Manitou"), Execute, Constants.MenuType_TwoWay, clsSvrFunctions.enCategory.enTwoWay);

AddParameters("CallKey", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter key for the call"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("WaitSecs", null, "60", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter the maximium time in seconds to wait"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**TwoWayRegisterCallSession**", clsGlobals.s_Language.GetText("Register Two-Way Call with Manitou"), Execute, Constants.MenuType_TwoWay, clsSvrFunctions.enCategory.enTwoWay);

AddParameters("CallKey", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter key for the call"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**TwoWayUnregisterCurrentSession**", clsGlobals.s_Language.GetText("Unregister Two-Way Call Session with Manitou"), Execute, Constants.MenuType_TwoWay, clsSvrFunctions.enCategory.enTwoWay);

Add("**TwoWayWaitForManitouNotifications**", clsGlobals.s_Language.GetText("Wait for Manitou to send a notification"), Execute, Constants.MenuType_TwoWay, clsSvrFunctions.enCategory.enTwoWay);

AddParameters("WaitSecs", null, "60", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter the maximium time in seconds to wait"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**TwoWayWaitForReceiver**", clsGlobals.s_Language.GetText("Wait for Receiver to pass Two-Way call"), Execute, Constants.MenuType_TwoWay, clsSvrFunctions.enCategory.enTwoWay);

AddParameters("WaitSecs", null, "60", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter the maximium time in seconds to wait"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**WaitForReceiver**", clsGlobals.s_Language.GetText("Wait for Receiver to finish call"), Execute, Constants.MenuType_ChannelBank, clsSvrFunctions.enCategory.enTwoWay);

AddParameters("WaitSecs", null, "60", enVariantType.Integer, clsGlobals.s_Language.GetText("Enter the maximium time in seconds to wait"));

Add("**TwoWayWaitForDisconnectFromTransfer**", clsGlobals.s_Language.GetText("Two-Way Wait for Operator to disconnect from Transfer"), Execute, Constants.MenuType_TwoWay, clsSvrFunctions.enCategory.enTwoWay);

Security M.F.C.s

Add("**DoesUserHaveCustomerAccess**", clsGlobals.s_Language.GetText("Does User Have Access to Customer"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enSecurity);

Add("**IsRegisteredCallerID**", clsGlobals.s_Language.GetText("Is Caller ID Registered"), Execute, Constants.MenuType_OpenVoice, clsSvrFunctions.enCategory.enSecurity);

AddParameters("CallerID", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter Caller ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**RegisterNewUser**", clsGlobals.s_Language.GetText("Register security details for new user"), Execute, Constants.MenuType_OpenVoice | Constants.MenuType_AutoText, clsSvrFunctions.enCategory.enSecurity);

AddParameters("CallerID", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Caller ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("MediaType", null, "0", enVariantType.String, clsGlobals.s_Language.GetText("Enter Media type (0=OpenVoice, 1=AutoText)"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("User", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter Manitou user ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Password", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter Manitou password"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("ExtUserId", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter new user ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("ExtUserPasswd", null, null, enVariantType.String, clsGlobals.s_Language.GetText("Enter new password"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("ExpireDays", null, 0, enVariantType.Integer, clsGlobals.s_Language.GetText("Enter number of days until expire"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

SoloProtect M.F.C.s

Add("**SendAudioSliceSignal**", clsGlobals.s_Language.GetText("Send Sliced Audio Signal"), Execute, Constants.MenuType_SoloProtect, clsSvrFunctions.enCategory.enSoloProtect);

AddParameters("TxID", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter transmitter ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("EvType", null, "SYS", enVariantType.String, clsGlobals.s_Language.GetText("Enter event type"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("StCode", null, "*A", enVariantType.String, clsGlobals.s_Language.GetText("Enter standard event code"), "", false,

clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Area", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter area"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Zone", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter zone"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("Line", null, "", enVariantType.String, clsGlobals.s_Language.GetText("Enter line number"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

AddParameters("PointID", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Enter point ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

Add("**WaitForAudioSlice**", clsGlobals.s_Language.GetText("Wait for Sliced Audio"), Execute, Constants.MenuType_SoloProtect, clsSvrFunctions.enCategory.enSoloProtect);

AddParameters("CallerID", null, "{$CID#}", enVariantType.String, clsGlobals.s_Language.GetText("Caller ID"), "", false, clsSvrFunctionParmameter.enModifier.enNothing);

SocialMedia M.F.C.s

Add(""**SendSignalSocialMedia**"", clsGlobals.s_Language.GetText(""Send a Signal for Social Media""), Execute, Constants.MenuType_Any, clsSvrFunctions.enCategory.enSocialMedia);"

AddParameters(""TxID"", null, ""{$SMTXID#}"", enVariantType.String, clsGlobals.s_Language.GetText(""Enter transmitter ID""), """", false, clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""EvType"", null, ""SYS"", enVariantType.String, clsGlobals.s_Language.GetText(""Enter event type""), """", false, clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""StCode"", null, ""*SMA"", enVariantType.String, clsGlobals.s_Language.GetText(""Enter standard event code""), """", false, clsSvrFunctionParmameter.enModifier.enNothing);"

```
AddParameters(""Area"", null, """", enVariantType.String,
clsGlobals.s_Language.GetText(""Enter area""), """", false,
clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""Zone"", null, """", enVariantType.String,
clsGlobals.s_Language.GetText(""Enter zone""), """", false,
clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""SearchString"", null, ""{$SMSS#}"", enVariantType.String,
clsGlobals.s_Language.GetText(""Enter search string""), """", false,
clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""Comment"", null, ""{$SMTXT#}"", enVariantType.String,
clsGlobals.s_Language.GetText(""Enter Comment""), """", false,
clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""FromID"", null, ""{$SMFUN#}"", enVariantType.String,
clsGlobals.s_Language.GetText(""Enter Social Media Screen ID""), """", false,
clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""URL"", null, ""{$SMSRC#}"", enVariantType.String,
clsGlobals.s_Language.GetText(""Enter URL for source""), """", false,
clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""URLInfo"", null, ""{$SMSRC#}"", enVariantType.String,
clsGlobals.s_Language.GetText(""Enter Description Info for source""), """", false,
clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""Latitude"", null, ""{$SMLAT#}"", enVariantType.String,
clsGlobals.s_Language.GetText(""Enter Latitude""), """", false,
clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""Longitude"", null, ""{$SMLON#}"", enVariantType.String,
clsGlobals.s_Language.GetText(""Enter Longitude""), """", false,
clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""RankZone"", null, true, enVariantType.Boolean,
clsGlobals.s_Language.GetText(""Rank Zone in 10's (43=40, 67=60) ""), """", false,
clsSvrFunctionParmameter.enModifier.enNothing);"

AddParameters(""SendProfile"", null, true, enVariantType.Boolean,
clsGlobals.s_Language.GetText(""Send Profile Image""), """", false,
clsSvrFunctionParmameter.enModifier.enNothing);"
```

Index