TAVIS™ Concentrator 2.2 – SDK Suite

User Manual

Document Revision 1.01

UM-C2031-R101-20060215

RF CODE

TAViS

Contents

TAVIS™ Concentrator 2.2 – SDK Suite

Appendix

Document History

Document Revised	Document ID
15 February 2006	UM-C2031-R101-20060215

Issue Date	Rev.	Comment
29 Aug 2005	00	Initial Draft/Release. 66 pgs.
21 Oct 2005	01	Revision – Applied design changes. 65 pgs.
15 Feb 2006	1.01	Revision – Design change and edits. 66 pgs.

Note

This document is not intended for public distribution. Copying of this material is not permitted without prior written permission.

1 – Introduction

What is the SDK Suite?

The TAVIS™ Concentrator 2.2 - SDK Suite by RF Code is composed of software components and processes that manage the collection of data from RFID readers. Utilizing a TCP/IP (Ethernet), serial, USB or other interface, the Concentrator supports all types of Auto-ID devices. The Concentrator provides users with the ability to construct small-embedded systems with one device to large systems consisting of hundreds of devices worldwide.

The TAVIS™ Concentrator 2.2 - SDK Suite is an application program interface (API) which enables a loosely-coupled network to perform configuration, monitoring and data collection on a wide variety of devices. A broad range of supported Auto-ID device types includes barcode, active RFID, passive RFID, RTLS, and GPS Location.

Consequently, devices manufactured by any company can be seamlessly integrated in one data-collection software system. The Concentrator driver system accomplishes this by dynamically attaching to devices provided by the device manufacturers. The Concentrator runs as a system service that is responsible for extracting information from data collection devices. The data can be extracted from the buffering system by using a single command.

Important – For TAVIS Concentrator system requirements and installation, please refer to the TAVIS Suite – XPlatform Installation & Suite Configuration user guide.

This manual is designed to help the user configure devices, set operating parameters, and interface with the Concentrator. This manual also provides information that will help facilitate the following Concentrator processes:

- Acquiring and deciphering status, alert, error and warning messages.
- Collecting, storing, and transferring data.
- Propagation and processing of commands.
- Time synchronization.
- Security.

Note – For more information on the TAVIS Concentrator GUI, please refer to the TAVIS Concentrator GUI user manual.

2 – System Configuration

What are the default port settings?

The TAVIS™ Concentrator 2.2 application contains the tavisConfig.bak file in its Settings directory. With a Windows INI type format, this file contains the default port settings to configure the Concentrator Service in the event that the configured ports are in use.

Note – Driver System Manager (DSMGR) commands are also provided for the import and/or export of the current system configuration. For more information on these commands, refer to page 10.

```
[Defaults]
DSMGRPort=7774
AlertServerPort=7776
EventExecPort=7778
ScriptPort=7790
RFLSPort=60011
ConPortDataServer=6700
ConPortServer=59001
ConPortClient=60002
ConPortServBuff=51999
```

If the tavisConfig.bak file has been edited, then the file must be renamed tavisConfig.ini in the same directory. Upon restarting the Concentrator Service, (1) this file will be read, (2) the port changes will be processed, and (3) the file will be renamed tavisConfig.bak once more.

3 – Process Communication

What is the Process Communication?

The TAVIS™ Concentrator 2.2 suite enables a command-level interface for process configuration and setup. Each process that incorporates this interface provides the following two commands – Echo and Strip – to allow customization of the command response level.

Command-level message responses have two distinct formats:

- ServiceName and Message – For simple commands and commands that return errors.
  - The ServiceName attribute represents the Process ID.
  - The Message attribute is the ID of the response.
- CommandName – For successful commands.

Note – For more information on Message Number assignments, refer to page 56, Appendix G.

What are the commands?

Note – Each command must end with a <CR> carriage return and must be comma-delimited (with no spaces before and after the comma) if they require arguments.

Echo

The Echo command allows the user to enable/disable command-level responses. By default, Echo is disabled (Off or 0).

Syntax	**Echo,On(1) | Off(0)<CR>**
Responses	For Strip 1, 3, and 4: **<Msg MsgType="Response" ServiceName="102" Message="0"/>** For Strip 2: **102,0** This correlates to **ServiceName,Message.** Refer to Message Number Assignments on page 56, Appendix G.
Examples	**Echo,On<CR>** **Echo,Off<CR>** **Echo,0<CR>** **Echo,1<CR>**

Strip

Strip

The Strip command allows the user to modify the data format for command-level responses. By default, the Strip format is set to 2.

Syntax	**Strip,1|2|3|4<CR>**
Examples	**Strip,3<CR>** **Strip,4<CR>**

See the following sample formats according to Strip value.

- Strip,1 – XML with Elements
- Strip,2 – Concise, comma-delimited
- Strip,3 – XML with concise records wrapped in <ER>
- Strip,4 – XML with attributes wrapped in <ER>

Strip Mode	1
Description	The <ER> tag encloses each record. Within the record, each field value is enclosed by a tag named with the field name. The <Table> tag may hold an optional name attribute.

Format

```
<Msg MsgType="Response" CommandName="Get">
  <Table>
    <ER>
      <ComponentName>Dev1</ComponentName>
      <DeviceType>Mantis</DeviceType>
      ...
    </ER>
    <ER>
      <ComponentName>Dev2</ComponentName>
      <DeviceType>Mantis</DeviceType>
      ...
    </ER>
  </Table>
</Msg>
```


Strip Mode	2
Description	Each record is delimited by a <CR> carriage return. Within the record, each field is comma-delimited. Strings must be enclosed in quotes.
Format	
"Dev1","DeviceType=Mantis","DeviceVersion=1.03",0,12<cr> **"Dev2","DeviceType=Mantis","DeviceVersion=1.03",1,5<cr>** **...**	

Strip Mode	3
Description	This is essentially strip,2 wrapped in XML, where the <ER> tag encloses each record. See strip,2 for field format.
Format	
<Msg MsgType="Response" CommandName="Get"> **<Table>** **<ER>”Dev1,”DeviceType=Mantis”,”DeviceVersion=1.03”</ER>** **<ER>”Dev2”,”DeviceType=Mantis”,”DeviceVersion=1.03”</ER>** **</Table>** **</Msg>**	

Strip Mode	4
Description	The <ER> tag encloses each record. Within the record, each field is an attribute.
Format	
<Msg MsgType="Response" CommandName="Get"> **<Table>** **<ER ComponentName="Dev1" DeviceType="Mantis" .../>** **<ER ComponentName="Dev2" DeviceType="Mantis" .../>** **</Table>** **</Msg>**	

4 – Driver System Manager

What does the DSMGR do?

All services are monitored for continuous operation. In case one fails, the Driver System Manager (DSMGR) process attempts to regain an operational state.

When the Primary Concentrator Service starts, it initiates and monitors the DSMGR. In turn, the DSMGR initiates and monitors all essential services, and can be used to start and stop individual Concentrator processes. Once the Primary service detects that the DSMGR has failed, it re-initiates the DSMGR and the cycle continues again.

Additionally, the Primary Concentrator Service and the DSMGR maintain reciprocal monitoring of each other. These processes are specified in the configuration file rconcmaster.ini under the [DSMGR] section, with the following defaults:

```
[DSMGR]
s1=Command Interface
s2=Device Manager
s3=Alert Agent
s4=Search Agent

[Command Interface]
description=description for process
run=program to run
```

The sN variables s1 through s4 specify the essential processes that will be initiated by the DSMGR. In fact, the DSMGR can manage a maximum of 10 processes.

The DSMGR accepts commands on the dsmgrport port, as set using the Command Interface. The following example sets dsmgrport to 7774.

```
set,system,dsmgrport,7774
```


What are the import/export commands?

The following DSMGR commands are provided for the import and/or export of the current system configuration. Key is a user-defined string (including Computer Name, Date, and Time, but must not include the invalid characters \ / : * ? " < > |) used to define the export file.

Command	Description
Export,Key	This creates an XML backup configuration of the current system for the associated key.
Size,Key	This gets the size of the file for the associated key.
Get,Key	This receives the XML backup for the associated key.
Import,size	To restore the system configuration, this command must be followed by the configuration information, as retrieved from the saved XML backup. Size is the size of the file that is about to be transmitted into DSMGR.

Upon import, DSMGR will stop all processes, import the new configuration, and then restart the processes.

What are the main commands?

Start

The Start command starts an individual process.

Syntax	**Start,processname<CR>**
Responses	OK – Process is marked to start. Use the Status command to obtain operational status. Already Started – Process is already started. No Such Service – Process is not present.
Examples	**Start,Search Agent<CR>**
Restrictions	Processname must match the process name as specified in the configuration file, not the process identifier.

Stop

The Stop command stops an individual process.

Syntax	**Stop,processname<CR>**
Responses	OK – Process is marked to stop. Use the Status command to obtain operational status. Already Stopped – Process is already stopped. No Such Service – Process is not present.
Examples	**Stop,Search Agent<CR>**

Status

Status

The Status command returns the status of an individual process.

Syntax	**Status,processname<CR>**
Responses	Up – Process is running. Down – Process is not running. No Such Service – Process is not present.
Examples	**Status,Search Agent<CR>**

Export

Export

The Export command creates an XML backup configuration of the current system for the associated key. Key is a user-defined string (including Computer Name, Date, and Time, but must not include the invalid characters \ / : * ? " < > |) used to define the export file.

Syntax	**Export,Key<CR>**

Note – For more information on the DSMGR import/export commands, refer to page 10.

Size

Size

The Size command gets the size of the file for the associated key.

Syntax	**Size,Key<CR>**

Get

Get

The Get command receives the XML backup for the associated key.

Syntax	**Get,Key<CR>**

Import

Import

To restore the system configuration, the **Import** command must be followed by the configuration information, as retrieved from the saved XML backup. **Size** is the size of the file that is about to be transmitted into DSMGR.

Syntax	**`Import,size<CR>`**

Note – For more information on the DSMGR import/export commands, refer to page 10.

Restart

Restart

The **Restart** command allows the user to reset the listening port upon a successful modification in the Command Interface process for the **DSMGRPort** setting.

Syntax	**`Restart<CR>`**
Responses	None – Socket will be terminated.
Examples	**`Restart<CR>`**

5 – Command Interface

What does the Command Interface do?

Note – The TAVIS™ Concentrator 2.2 is dependent on the Command Interface. Other features of this application will not be available unless this process is running.

The Command Interface process allows the TAVIS™ Concentrator 2.2 to be remotely configured and controlled via a series of commands. It is also responsible for transmitting stored data from the buffering system. Acting as a server, this interface responds to commands and transmits data that is stored in the buffering system.

- Command Port – With a default TCP port number of 59001, this is the port (ConPortServer property) on which the Command Interface listens for and accepts commands.
  - For more information on the list of commands, refer to page 22.
  - The Command Interface must be restarted if the Command Port number is changed.
  - This port can accept up to ten connections. If the connection is not available, then the "<No more connections allowed/>" message will be sent.
- Data Port – With a default TCP port number of 51999, this is the port (ConPortServBuff property) through which the Command Interface streams buffer data from the buffering system, in response to the NextBufferD or NextBufferA command. In order to receive data through this port, there must be a connection established to both the Command Port and the Data Port.
  - For more information on reading buffer data, refer to page 20. For more information on NextBuffer commands, refer to page 30.
  - The Command Interface must be restarted if the Data Port number has been changed.
  - This port can accept only one connection. If the connection is already occupied, then the "<Connection in use/>" message will be sent.

How do I manage my devices?

Before devices can be used, they must first be added and configured in the system via the TAVIS™ Concentrator 2.2.

Note – Devices may also be added and configured via the TAVIS Concentrator GUI. For more information, refer to the TAVIS Concentrator – GUI user manual. To use the SDK Suite, refer to the instructions below.

Adding Devices

The TAVIS™ Concentrator 2.2 supports any RFID device with a driver that complies with the Device Driver Interface. The device properties are dynamically loaded from the device driver as soon as the device type becomes known to the system. When added to the system, several properties are associated with the device:

- DeviceName – This property is a value provided by the user when adding a new device.
- StreamOut – This property indicates if the device data should be provided to the DirectStream port.
- DeviceDriver – This property loads the specific driver. Once loaded, the driver can provide additional properties. For more information on device types, refer to page 48, Appendix A.

Note – The maximum memory allocated for the properties of each device is 10 KB. (Future versions of the Concentrator will dynamically allocate memory for the device properties.)

To add a device to the system:

Step 1 – Add the device name using the Add command (see page 25).

Step 2 – Set the DeviceDriver property by using the Set command (see page 23).

- Example – The following example uses the Add and Set commands to add a device and set its device properties.

```
Add,Dev1
Set,Dev1,DeviceDriver,devicedrivername
```

- To obtain the values for the devicedrivername, use the Get command below.

```
Get,System,devicetypes
```


Configuring Devices

After the DeviceDriver property is set, all of the device specific properties will be loaded into the system. At this point, the user will be ready to read from and/or write to these properties.

- Refer to the device vendor documentation for information concerning the device property values.
- Use the Get and Set commands to read and write to the device properties, respectively.
- Example – The following example uses the command to set the properties necessary to start an RF Code Mantis™ RFID device.

```
Set,Dev1,ConnectionMode,10.193.1.230
Set,Dev1,ServerPort,6500
```

Deleting Devices

Note – The device must be stopped prior to attempting to delete the device.

To remove a device and all of its settings, send the Delete command.

- Example – The following example uses the command to remove the devices named Dev1 and Dev2.

```
Delete,Dev1
Delete,Dev2
```

Getting Device Information

To list the device information, use the Get,DevN command.

- Example – The following example uses the command to list the device property information for a device called Dev1.

```
Get,Dev1,All
```


- The following is a sample response in Strip,1 format (for Strip formats, see page 7).

```
<Msg MsgType="Command" CommandName="Get">
 <Table>
  <ER>
   <ComponentName>Dev1</ComponentName>
   <StreamOut>1</StreamOut>
   <DeviceDriver>Mantis (network)</DeviceDriver>
   <DeviceType>Mantis</DeviceType>
   <DeviceVersion>(unknown)</DeviceVersion>
   <ConnectionMode> </ConnectionMode>
   <ServerPort>6500</ServerPort>
   <BaudRate>0</BaudRate>
   <RegMode> </RegMode>
   <DeviceMode>20</DeviceMode>
   <MaxRange>8</MaxRange>
   <MinRange>8</MinRange>
   <Range>8</Range>
   <RangeTime>60</RangeTime>
   <FactorMethod>standard</FactorMethod>
   <TagList> </TagList>
   <TagGroup>RFCDAA</TagGroup>
   <RangeFactor>1</RangeFactor>
   <DeviceExec> </DeviceExec>
   <MaxAutoReconnects>100</MaxAutoReconnects>
   <ReconnectInterval>10</ReconnectInterval>
   <CheckStatusInterval>5</CheckStatusInterval>
   <MaxStatusFail>5</MaxStatusFail>
   <WatchDogTime>5</WatchDogTime>
   <TagTimeout>60</TagTimeout>
   <SSI>0</SSI>
   <TimeStamp>0</TimeStamp>
   <DriverVersion>0001</DriverVersion>
  </ER>
 </Table>
</Msg>
```


Starting & Stopping Devices

Note – A device can be started only after it has been successfully added and configured. When a device is stopped, all tag data read by the device is purged from the Concentrator.

To start and stop a device, use the Start and Stop commands, respectively.

- Example – This uses the Start command to start Dev1 and Dev2.

```
Start,Dev1
Start,Dev2
```

- Example – This uses the Stop command to stop Dev1 and Dev2.

```
Stop,Dev1
Stop,Dev2
```

Monitoring Devices

The state of a device can be retrieved by sending the Get,Device,Status command to the Communications Agent.

- Example – This uses the command to send back the state and the connection state of device Dev1.

```
Get,Dev1,status
```

- Example – This is a sample response in Strip,2 format (for Strip formats, see page 7).

```
“Dev1”,“Status=4”
```

Note – For more information on component status codes, refer to page 49, Appendix D.

Note – The Get,Devices,Status command will respond with statuses for all devices. The output will follow according to the current Strip format.

How do I manage my device data?

Buffering

The Concentrator transfers received device data from memory buffers to disk. When the Concentrator receives tag data from the device and buffering is enabled (refer to DBOutput on page 50, Appendix E), it copies it into buffer memory. When buffering is disabled, no buffer file is created, but the data may still be sent via Direct Stream (see page 21).

When a maximum number of records are reached or a maximum time has expired, whichever occurs first, the Concentrator saves the buffer memory to disk. The two properties are MaxNumRecord (Buffer Record Limit) and MaxTime (Buffer Time Limit).

Buffer File Creation

The file name is generated, based on a counter. Once a counter limit is reached, the Concentrator begins to reuse files, replacing the oldest with new ones. The current file name is kept in a special file to maintain persistent naming.

- Example – This sets the maximum time to 50 seconds.

```
Set,System,MaxTime,50
```

- Example – This sets the maximum number of records to 100.

```
Set,System,MaxNumRecord,100
```

- Example – This retrieves the buffer values.

```
Get,System,MaxTime
Get,System,MaxNumRecord
```

- Example – This sets the buffer delete mode to Manual.

```
Set,System,BufferIntegrity,1
```


To conserve disk space, the DBOutput property may be set to 0, which disables buffering.

- Example –This disables buffering.

```
Set,System,DBOutput,0
```

- Example – This enables buffering.

```
Set,System,DBOutput,1
```

Reading Device Data

There are two ways to obtain device data as follows.

- Data Server Port
  - Specified by the ConPortServBuff property, this port is used for data transfers from the Concentrator. To read data from a buffer, connect a socket to this port, then send a NextBufferD to the Command Port of the Command Interface. In response to this command, a single buffer will be sent to the Data Server port. For more information on NextBufferD, refer to page 31.
  - To view all data from buffers, a NextBufferD must be repeatedly sent until the Empty response message is received (102,[00000]10006). For more information on Message Number assignments, refer to page 56, Appendix G.
- Direct Stream
  - Specified by the ConPortDataServer property, the Direct Stream port is another way of getting device data. The Concentrator Service provides data on port 6700 by default. Once the client connects to this port, the Concentrator sends the device data to the client formatted as RFCP (RF Code Common Protocol) data.
  - A maximum of ten clients can connect to the port simultaneously.

Notes

- The output socket is non-blocking, which means that if the client does not read fast enough, a client might lose data. If the Concentrator is unable to send to this socket, due to a full socket buffer, the RFCP data will be lost.
- Sending the CloseDataConnection command to the Data Server port ensures the release of the opened socket.

What are the main commands?

Note – Each command must end with a <CR> carriage return and must be comma-delimited (with no spaces before and after the comma) if they require arguments.

Strip Filter

Strip Filter

There are several filters that can be used for the NextBufferD and NextBufferA commands. For more information on NextBuffer commands, refer to page 30.

Syntax	**Strip,StripMode[NextBufferFilter]<CR>**
Examples	**Strip,12<CR>** (Strip 1, Filter 2) **Strip,42<CR>** (Strip 4, Filter 2)

The filter mode (value) is the second digit of the Strip command parameter.

- See the table below for filter descriptions according to filter value.

Filter#	Description
<none>	No filter.
1	Filters for unique tags.
2	Filters for only the "H" event code. For more information on event code descriptions for the NextBufferD command, refer to page 31.
3	Filters for only the "M" event code. For more information on event code descriptions for the NextBufferD command, refer to page 31.
4	Filters for tags found at the minimum range.
5	Filters for tags found at the maximum range.

Note – In a future release, this feature will be extended by providing separate Strip and Filter properties. This is to extend the current limitation of only 10 strips and 10 filters.

Set

The Set command sets the property of the device specified by the device name or the system.

Syntax	**SET,COMPONENT,PROPERTY,VALUE** **COMPONENT ::= ["SYSTEM" | DeviceName] ~"NONE"** **"SYSTEM"** – SYSTEM COMPONENT. **DeviceName** – Individual device (RFID device). **~NONE** – Reserved name. **PROPERTY** – Property name in component. **VALUE** – Value specific to component property.
Responses	OK – Property was successfully set. Or another message describing the reason for failure.
Examples	**Set,System,MaxTime,100** **Set,Dev1,ConnectionMode,10.193.1.101**

Note – See the individual device documentation for the list of properties.

Get

Get

The Get command retrieves the property (or properties) of the component specified by the component name.

Syntax	**GET,COMPONENT,PROPERTY** **COMPONENT ::= ["SYSTEM" | Devices | DeviceName] ~"NONE"** **"SYSTEM"** – SYSTEM COMPONENT. **Devices** – All devices (RFID devices) in system. Not fully supported at this time; Status, DeviceName, DeviceDriver, and StreamOut properties only. **DeviceName** – Individual device (RFID device). **~NONE** – Reserved name. **PROPERTY := [PropertyName | All | Status]** **PropertyName** – Property name in component. **All** – All property names in component(s). **Status** – Provide status information on Component. For more component status values, refer to page 49, Appendix D.
Responses	Command: **Get,Dev1,ConnectionMode** Response: **"Dev1","ConnectionMode=10.193.1.214"**
Examples	**Get,System,MaxTime** **Get,Dev1,ConnectionMode**

Note – See the individual device documentation for the list of properties.

Add

Add

The Add command adds a new device to the Concentrator.

Syntax	**Add,DeviceName<CR>**
Responses	OK – Device was added. No Space – No more devices can be added. Name Too Long – DeviceName exceeds 50 characters. Name Conflict – A device with the same name already exists.
Examples	**Add,Dev1<CR>**
Restrictions	DeviceName cannot be in the Reserved Device Names list (see page 49, Appendix B), or larger than 50 characters.

Delete

Delete

The Delete command removes a device from the database tables.

Syntax	**Delete,DeviceName<CR>**
Responses	OK – Device was successfully removed. No Such Device – Specified device does not exist. Cannot Remove Started Device – Device must be stopped before attempting to delete it.
Examples	**Delete,Dev1<CR>**
Restrictions	DeviceName cannot be in the Reserved Device Names list (see page 49, Appendix B), or larger than 50 characters.

Start

Start

The Start command starts one device or all devices.

Syntax	**Start,DeviceName | All<CR>**
Responses	OK – Device is ready to start and is marked to be started. However, this does not mean that the device is guaranteed to be started. There could be other problems (e.g. no connection, network down, power outage, etc.). Use the GET,DEVICENAME,STATUS or Alert messages to diagnose the problem. No Device or Already Started – Device is not added to the system yet or it is already started. No Service – Service is not present.
Examples	**Start,Dev1<CR>**
Restrictions	DeviceName cannot be in the Reserved Device Names list (see page 49, Appendix B), or larger than 50 characters.

Stop

Stop

The Stop command stops one device or all devices.

Syntax	**Stop,DeviceName | All<CR>**
Responses	OK – Device was started and it was marked to be stopped. However, this does not mean that the device is guaranteed to be stopped. There could be other problems (e.g. no connection, network down, power outage, etc.). Use the GET,DEVICENAME,STATUS or Alert messages to diagnose the problem. No Device or Already Stopped – Device is not added to the system yet or it is already stopped. No Service – Service is not present.
Examples	**Stop,Dev1<CR>**
Restrictions	DeviceName cannot be in the Reserved Device Names list (see page 49, Appendix B), or larger than 50 characters.

GetTime

GetTime

The GetTime command sends back the local system time. The format of the date and time (in military time) is displayed according to the operating system settings.

Syntax	**GetTime<CR>**
Responses	**6/24/2004 12:35:00 PM**
Examples	**GetTime<CR>**

SetTime

SetTime

The SetTime command sets the date and time (in military time) of the operating system.

Syntax	**SetTime,mm/dd/yyyy hh:mm:ss<CR>**
Examples	**SetTime,01/01/2003 17:35:00<CR>**

Reboot

Reboot

The Reboot command will reboot the Concentrator computer. It will not send a response.

Syntax	**Reboot<CR>**
Examples	**Reboot<CR>**

Purge

Purge

The Purge command sends a request to the Concentrator Service to remove all data buffer files that it has created. If Data Buffering (NextBufferD) was enabled while processing this command, it is turned off. Once the process is completed, Data Buffering is turned back on.

Syntax	**Purge<CR>**
Responses	OK – All data buffer files were deleted successfully. No Service – Concentrator is not running.
Examples	**Purge<CR>**

Note – The Purge command acts upon files in the local file system. The time to process depends on the number of files in the data directory.

Store

Store

The Store command works the same way as the Import command in DSMGR. The parameters consist of the following:

- Key is actual name for the file, without path elements, with the same constraints that are used in the devicename.
- Size is the size of the buffer that is about to be streamed in on the Store command.
- Attribute is an integer value. Currently, three values are available:
  - 0 = Configurator.
  - 1 = Host Registration – Spider host registration files.
  - 2 = Scripts – New script engine files.

Syntax	**Store,key,attribute,size\r**

Note – For more information on DSMGR import/export commands, see page 10.

Retrieve

Retrieve

The Retrieve command works the same way as the Export command in DSMGR. The parameters consist of those available for the Store command.

Syntax	**Retrieve,key,attribute\r**

Note – For more information on DSMGR import/export commands, see page 10.

Size

The Size command gets the size of the file for the associated key. The parameters consist of those available for the Store command. All responses are in Strip,1 format.

Syntax	`Size,key,attribute\r`
Example	`size,x,2\r`

Response	`<Msg MsgType="Response" CommandName="size">`
	`  <Table>`
	`  <ER><Size>483</Size></ER>`
	`  </Table>`
	`</Msg>`

List

The List command gets the information for a specific key or all keys (*).The parameters consist of those available for the Store command. All responses are in Strip,1 format.

Syntax	`List,key,attribute\r`
Example	`list,*,2\r`

Response	`<Msg MsgType="Response" CommandName="list">`
	`  <Table>`
	`  <ER>`
	`  <Key>keyx</Key>`
	`  <Size>486</Size>`
	`  <ComponentName>XName</ComponentName>`
	`  </ER>`
	`  <ER>`
	`  <Key>keyy</Key>`
	`  <Size>483</Size>`
	`  <ComponentName>YName</ComponentName>`
	`  </ER>`
	`  </Table>`
	`</Msg>`

StartScript StartScript

The StartScript command interacts with the Script Engine to begin Script processing on the specified key.

Syntax	**StartScript,keyx\r**
Example	**StartScript,Script1<CR>**
Response	**102,0,KEY<CR>**

StopScript StopScript

The StopScript command interacts with the Script Engine to terminate an active Script processing on the specified key.

Syntax	**StopScript,keyx\r**
Example	**StopScript,Script1<CR>**
Response	**102,0<CR>**

ActiveScript ActiveScript

The ActiveScript command interacts with the Script Engine to retrieve a listing of all active Scripts or a specific key.

Syntax	**ActiveScript,All | keyx\r**
Example	**ActiveScript,Script1<CR>**
Response	**<Msg MsgType="Response" CommandName="Active"><Table><ER><KEY>Script1</KEY></ER></Table></Msg>**

What are the NextBuffer commands?

When the NextBufferD or NextBufferA command is received, data is sent through the Data Server Port. With a default TCP port number of 51999, this port can accept only one connection.

CloseDataConnection

CloseDataConnection

The CloseDataConnection command is sent to the Data Server Port to ensure the release of the opened socket.

Syntax	**CloseDataConnection<CR>**
Examples	**CloseDataConnection<CR>**

DeleteBuffer

DeleteBuffer

The DeleteBuffer command is used to manually delete the buffer only when the BufferIntegrity property is set to 1 (Manual Delete).

Syntax	**DeleteBuffer<CR>**
Examples	**DeleteBuffer<CR>**

- Example – The following example sets the buffer delete mode to Manual.

```
Set,System,BufferIntegrity,1
```

Note – For more information on the BufferIntegrity system property, see page 50, Appendix E.

NextBufferD

NextBufferD

The NextBufferD command fetches data records from the file buffering system to the Data Server Port.

Syntax	**NextBufferD<CR>**
Examples	**NextBufferD<CR>**

When there is no more data from the tables, an empty message is sent.

- To view all data from buffers, a NextBufferD must be repeatedly sent until the Empty response message is received (102,[00000]10006). For more information on Message Number assignments, see page 56, Appendix G.
- For data Strips 1, 3, and 4 (see page 7), the message is in the following format:

```
<Msg MsgType="Response" ServiceName="102"
   Message="10006"/>
```

- For data Strip 2, the message is formatted as a comma-delimited list.

```
EventID,TagID,Range,DeviceName,EventTime,TagEvent,
   GroupCode, EventData1,EventData2,EventInfo,
   RecordInfo<CR>

Example:
99,"00009928",1,"Mantis227","08/16/03 04:30:27 PM","H",
   "HITACH",Relay1=777, , ,
```

- See the following data response descriptions for Strip 2.

Data Response Descriptions – Strip 2

Data Field	Description
EventID	The EventID is a sequential number (counter) assigned by the application and is incremented with each record that is recorded. The Event ID is unique for each Concentrator and ranges from 1 to 2,147,483,647 before it starts over. The Event ID is reset to 1 upon reboot. This event counter applies only for data, not for messages of any type. For more information on message formats, refer to page 43.
TagID	The TagID is the alphanumeric ID for tag identity.
Range	The Range number is a relative integer number (starting at 1) that is reported for tag distance, as delivered by the reading device (where lower numbers indicate that the tag is closer to the reading device). These readings can provide attenuation or SSI readings or any other measurement that the device provides for distance.
DeviceName	The DeviceName defines the reader (or antenna) that reads the tag identity that created this record.
EventTime	The EventTime gives the date and time stamp when the returned data record was recorded. The acceptable time formats are as follows: **mm/dd/yy hh:mm:ss [AM|PM]** (AM or PM) **mm/dd/yyyy hh:mm:ss** (Military time) **mm/dd/yy hh:mm:ss** (Military time)
TagEvent	The TagEvent defines the residency status of the TagID in the device that sent the ID. For more information on the defined TagEvent Field Codes, refer to the table on page 33 below.
GroupCode	The GroupCode defines the category, to which the Tag ID belongs in a regular tag event such as group code or customer ID (i.e. an event that is not identified as "X", "Y", or "Z" in TagEvent).
EventData1	This field is used for sensor data as follows: **SSI=3;Temp=90;Relay="RF ActivatorID";** **DeviceType="Device type name"** This gives the option to provide information of known keywords for next stage processing. If there is more than one value for a specific reading, then they will be enumerated and separated by a semi-colon (;). For example, for two channels, the SSI reading will appear as follows: **SSI1=99;SSI2=78** Note – If the TagEvent code is "P", then the EventData1 field will contain the pointer ID in the following format: **Pointer="Pointer ID"**

Data Field	Description
EventData2	This additional field provides a placeholder for full XML files and documents that describe tag events data or any data being sent through the Concentrator devices. This field is activated through the EventInfo. In other words, one will check these fields for information only if TagEvent is "X", "Y", or "Z", and the "TypeInfo" information has the value "XML" in EventData1.
EventInfo	The EventInfo is used to describe that type of data that is in fields EventData1 and EventData2. The information is formatted as follows: **EventData1:Keyword="Description";** **EventData2:Keyword="Description"** In case of "X", "Y", or "Z" event, this field can provide scripts for additional data formatting, unit conversions, or data interpretation of information from new devices. In case of a script, the EventInfo field contains the following string: **%Type="Script Type" Lang="Script Language and Version" Script="The actual script"**
RecordInfo	The RecordInfo provides a link for additional information in the next record and hold the values as follows: **ContinueOnEvent=112222** (and the next one will be) **ContinuePart1=112221** This way multiple records could be concatenated (linked) to provide more information for future needs.

TagEvent Field Code Descriptions

Event	Description
H	In Continuous mode, "H" represents a "hit," which occurs every time the TagID is being transmitted or interrogated in a usual behavior of the reading device.
N	In Exception mode, "N" represents "new". This value indicates the TagID has been newly found by the reader
L	This event in Exception mode means that TagID had been resident with the reading device, but is now lost.
M	The TagID is being transmitted because of motion.
T	The TagID is being transmitted because of tamper.
Te	The TagID is being transmitted because of the threshold or device interrogation. The temperature value may be obtained in the EventData1 field under the "Temp" keyword.
R or **H**	The TagID is being transmitted because of RF activation. Relayed information may be obtained in the EventData1 field under the "Relay" keyword.

The field code descriptions are continued below.

Event	Description
P	The TagID is being transmitted because of a pointer (panic). Relayed information of pointer ID may be obtained in the EventData1 field under the "Relay" keyword. Note – If the TagEvent code is "P", then the EventData1 field will contain the pointer ID in the following format: **Pointer="Pointer ID"**
I	The TagID is being transmitted because of infrared or photoelectric sensor. EventData1 keywords will "IR".
B	The TagID is being transmitted because of low battery threshold or device interrogation. The battery level value may be obtained in the EventData1 field under the "BATT" keyword.
S	The TagID is being transmitted because of an arbitrary sensor. The sensor information may be found in the EventData1 under the sensor name keyword and matching sensor functionality.
C	The TagID is barcode read information.
CA	The TagID is being associated with the barcode ID in the EventData1 field under the keyword "Barcode".
RM	The TagID is being transmitted because of RF activation and motion. Relayed information of pointer ID may be obtained in the EventData1 under the "Relay" keyword.
X	Identify that this event is not a regular tag event and the information is of a different type. As such, the user should treat the information in this event according to the keyword "InfoType" in the EventData1 field.
Y	Identify that this event is not a regular tag event and the information is of a different type. As such, the user should process the information in that event according to the script in EventInfo that starts with "Script:".
Z	Identify that this event is not a regular tag event and the information is of a different type. As such, the user should treat the information in this event according to the keyword "InfoType" in the EventData1 field, and process the information according to the Perl script in EventInfo that starts with "Script:"

NextBufferA NextBufferA

The NextBufferA command sends data, status and error records. After the data is transferred, the buffers are cleared.

Syntax	**NextBufferA<CR>**
Examples	**NextBufferA<CR>**

Note – This command is provided for backward compatibility only. The NextBufferD command is a better choice for new development.

What are the compatibility commands?

Note – The TAVIS™ Concentrator 2.2 commands are recommended for new development. Meanwhile, the following commands are provided for backwards compatibility, and may be removed in a future release.

Note – The available command equivalencies include the following:

- The AddD and RemoveD commands can be replaced by the Add and Delete commands.
- The List commands can be replaced by equivalent Get commands.
- The Update command can be replaced by a Set command.

Add & Remove Commands

AddD AddD

The AddD command adds a device to the Devices database table and updates the Properties values in the SysSetting table. The remaining parameters are inserted via the UpdateD and UpdateSS commands.

Syntax	**AddD,DeviceName<CR>**
Examples	**AddD,Spider2<CR>**

RemoveD RemoveD

The RemoveD command removes a device from the Devices database table.

Syntax	**RemoveD,DeviceName<CR>**
Examples	**RemoveD,Spider2<CR>**

List Commands

ListD

ListD

The ListD command lists a limited set of properties associated with all devices assigned to the Concentrator.

Syntax	`ListD,DeviceName | All<CR>`

Responses (Strip 1)	`<Msg MsgType="Response" CommandName="ListD">`
	`<Table>`
	`<ER>`
	`<ComponentName>Mantis1</ComponentName>`
	`<DeviceType>Mantis</DeviceType>`
	`<DeviceVersion>1.35</DeviceVersion>`
	`<ConnectionMode>10.193.1.50</ConnectionMode>`
	`<IsRunning>0</IsRunning>`
	`<Start>0</Start>`
	`<RefreshDev>0</RefreshDev>`
	`<RefreshSys>0</RefreshSys>`
	`</ER>`
	`</Table>`
	`</Msg>`

Examples	`ListD,Dev1<CR>`

Note – This command is provided for backwards compatibility only. The Get command (get,devices,all) is recommended for new development.

ListSS

ListSS

The ListSS command sends back the contents of the tblSys_Setting table.

Syntax	`ListSS,DeviceName|All<CR>`

Responses (Strip 3)	`<Msg MsgType="Response" CommandName="ListSS">`
	`<Table>`
	`<ER>dev,A,1,1,1,30,Standard,,RFCODE,20,,1,</ER>`
	`</Table>`
	`</Msg>`

Examples	`ListSS,Dev1<CR>` `ListSS,All<CR>`

ListSP

ListSP

The ListSP command lists the System Properties of the Concentrator.

Syntax	**ListSP<CR>**
Examples	**ListSP<CR>**

Note – This command is provided for backwards compatibility only. The Get command (get,system,all) is recommended for new development.

ListPP

ListPP

The ListPP command retrieves the Parent Parameters of the Concentrator, such as the IP address of a Control Station or a Collector.

Syntax	**ListPP<CR>**
Examples	**ListPP<CR>**

Note – This command is provided for backwards compatibility only. The Get command (get,system,all) is recommended for new development.

Update Commands

Update commands are the control access to the readers in the network, which allow the user to change the control parameters.

Note – The value parameter for each command must contain the appropriate delimiting characters according to the Jet SQL standard of SQL syntax. Each command must end with a <CR> carriage return and must be comma-delimited (with no spaces before and after the comma) if they require arguments.

UpdateSS

UpdateSS

The UpdateSS command updates the property values in the SysSetting table. These changes are not immediately processed by the operating system. To implement the desired changes in real-time, the user must follow the Update command with the applicable RefreshDevice or RefreshSys command.

Syntax	**UpdateSS,Device|All,Fieldname,Value<CR>**
Examples	**UpdateSS,All,RegMode,H<CR>** **UpdateSS,Dev1,Range,8<CR>**

UpdateSP UpdateSP

The UpdateSP command updates the property values in the SysParams table. These changes are not immediately processed by the operating system. To implement the desired changes in real-time, the user must follow the Update command with another command (Stop/Start, RefreshDevice or RefreshSys) depending on the situation.

Syntax	**UpdateSP,Fieldname,Value<CR>**
Examples	**UpdateSP,UseSoundNotify,Yes<CR>** **UpdateSP,MaxNumRecord,100<CR>**

UpdatePP UpdatePP

The UpdatePP updates the property values in the ParentParams table.

Syntax	**UpdatePP,Fieldname,Value<CR>**

UpdateD UpdateD

The UpdateD command updates the property values in the Devices_IP table.

Syntax	**UpdateD,Device|All,Fieldname,Value<CR>**
Examples	**UpdateD,All,RefreshDevice,Yes<CR>** **UpdateD,Spider1,Start,Yes<CR>**

How are error messages reported?

When error messages occur, they are sent to the Command Port of the Command Interface (see page 13).

- The format of the message is specified by the Strip command (see page 7).
- The error messages are only sent if Echo is enabled (see page 6). By default, Echo is disabled.

How does the IP-blocking security work?

The IP Blocking mechanism provides a level of security to prevent unauthorized access to the TAVIS™ Concentrator 2.2. In fact, the Concentrator is set so that the first client who connects to any service becomes the owner, and the only access is provided via the localhost (e.g. 127.0.0.1).

- Example – For example, if the first connecting client IP address is 10.193.1.101, then 10.193.1.101 becomes the master IP address. Consequently, 10.193.1.101 is the only IP address that can connect to any service of the Concentrator. From this point, the client at 10.193.1.101 can enter new IP addresses in the table, and block or unblock them by sending the proper commands to the Command server.

The connections to the TDCP (TAVIS™ Data Collection Platform) require administrative permission. That permission is granted via an IP-address Unblock command. By default, all addresses are blocked except the address specified at install time and the Concentrator GUI when used on the same machine as the Concentrator SDK.

To be unblocked, each address must be explicitly added to the database. In other words, only users on the local computer may edit the IP-blocking database to add, delete or modify security access.

- A maximum of 50 IP addresses can be held in the database.
- If an IP address is blocked, then the "<IP blocked/>" message will be sent.

To modify the IP-blocking database, the commands accepted by the Command Interface are as follows.

Command	Description
AddIP,<IP Address>	This adds an IP address to the database.
DeleteIP,<IP Address>	This deletes an IP address from the database.
BlockIP,<IP Address>	This blocks an IP address.
UnblockIP,<IP Address>	This unblocks an IP address.
ListIPS	This lists all of the IP addresses.

Note – Similar actions can be done with the Set and Get commands. For more information on system properties, refer to page 50, Appendix E.

Below is a more-descriptive list of the IP security commands.

Security Commands

AddIP

AddIP

The AddIP command adds an IP address to the security database.

Syntax	**AddIP,IPaddress<CR>**
Examples	**AddIP,10.193.1.87<CR>**

Note – A maximum of 50 IP addresses can be held in the database.

DeleteIP

DeleteIP

The DeleteIP command deletes an IP address from the security database.

Syntax	**DeleteIP,IPaddress<CR>**
Examples	**DeleteIP,10.193.1.101<CR>**

BlockIP

BlockIP

The BlockIP command blocks an IP address from connecting to the Concentrator.

Syntax	**BlockIP,IPaddress<CR>**
Examples	**BlockIP,10.193.1.34<CR>**

UnblockIP

UnblockIP

The UnblockIP command allows an IP address to connect to the Concentrator.

Syntax	**UnblockIP,IPaddress<CR>**
Examples	**UnblockIP,10.193.1.66<CR>**

ListIPs

ListIPs

The ListIPs command lists all IP addresses and their corresponding access statuses.

Syntax	**ListIPs<CR>**
Examples	**ListIPs<CR>**

6 – Alert Agent Service

What does the Alert Agent Service do?

The Alert Agent Service is an independent process that receives local alert messages from the TAVIS™ Concentrator 2.2 suite of processes. By default, it listens for incoming commands and alert messages on port 7776. This port setting can be modified by using the set,system,alertserverport command in the Command Interface process.

Notes

- Alert subscriptions can be added to the Alert Agent for notification and delivery to remote systems. A maximum of 50 subscriptions is allowed.
- The IP address of the client must be granted access as an incoming IP address. For instructions on IP access, refer to page 39.

What are the commands?

Alert

The Alert command enables the user to subscribe to specific diagnostic alerts and be notified in specified protocols.

Syntax	**Alert,Type,Action,Protocol,DestinationIP,DestinationPort [,SiteIP,SitePort]/** where **[Future Use]**
Responses	OK – Command executed successfully.
Examples	To subscribe for device state alerts, send the following command: **Alert,Device="*" State="*" Diag="None",Subscribe, TCP,10.193.1.50,5555<CR>** To subscribe for device noise alerts, send the following command: **Alert,Device="*" State="None" Diag="Noise",Subscribe, TCP,10.193.1.50,5555<CR>**

- See the following data formats for the protocol properties (parameters).

Data Formats for Protocol Properties

Property	Description
Type	Values: Device = "DeviceName" State = "State" Diag = "Noise" or "None" Notes: All three attributes are required and must be delimited by a space. For Device and State, the wildcard character (*), may be used to indicate "All". For State or Diag, the keyword "None" may be used. Device - This indicates the device (or component) for which the alert provides information. State – Refer to the Alert Event State table on page 49, Appendix C. Noise – This provides an alert if one of the noise levels exceeds threshold. The Noise alert is only generated if the device is capable of generating such an event.
Action	Values: "Subscribe" or "Unsubscribe" Notes: Either value is required. Subscribe - This subscribes an alert. Unsubscribe - This removes a subscription.
Protocol	Values: TCP Notes: This value is required.
DestinationIP	Values: IP address to send Alert. Notes: This value is required.
DestinationPort	Values: Port address to send Alert. Notes: This value is required.
SiteIP	Values: IP address for remote TAVIS™ Gateway. Notes: This value is optional. A Relay Agent forwards the Alert to IP:Port. The default is empty. For future use.
SitePort	Values: Port address for remote TAVIS™ Gateway. Notes: This value is optional. The default is empty. For future use.

How will the alert message be sent?

An alert is sent to a specified address (the DestinationIP and DestinationPort properties) on the network, unless it is configured to be sent over the Internet via the TAVIS™ Gateway.

- The alert message will include the EventID, SystemName, DeviceName, EventTime, TagEvent, EventData1, and EventInfo.
- The message will be sent according to the RFCP protocol described in the table below.

Notes

- When sending an alert over the Internet, the address of the remote TAVIS™ Gateway is placed in the SiteIP and SitePort properties. The TAVIS™ Gateway then forwards the alert to the subscriber on the other site.
- The SiteIP and SitePort properties are reserved for future use.

Alert Message Field Values

Data Field	Value
TagEvent	Refer to the Alert Event State table on page 49, Appendix C.
EventData1	Reconnection to device failed due to network error.
EventInfo	AlertType="Concentrator"

See the following sample alert in Strip 3 format.

Format

```
<Alert>
  <EventId>1914</EventId>
  <SystemName Type="Concentrator">Conc2</SystemName>
  <ComponentName>M131</ComponentName>
  <EventTime>09/25/03 06:09:08 PM</EventTime>
  <TagEvent Alert="Up"/>
  <EventData1>
    <AlertDescription>Reconnect attempt succeeded</AlertDescription>
    <Diag>None</Diag>
    <ip>10.193.1.244</ip>
    <port> 5555</port>
  </EventData1>
</Alert>
```

7 – SDK Scripting Agent

What does the SDK Scripting Agent do?

The SDK Scripting Agent provides the ability to send any SET command configured by the device. Along with current set of commands by RF Code, this agent allows for a robust enhancement of the TAVIS™ family.

This new scripting engine avoids the problem of implementing driver-side commands to control the operation of specific devices, which would entail numerous and tedious driver-side changes. Instead, a central point now controls the behavior based upon an XML script. The script engine is setup to read XML as server-side files or as passed-in XML streams.

The basics of the script is the Engine command <eng/>. This command tells the processor what the order number is for the command and what type of command we have. Commands can be setcmd, cmd and goto.

What are the commands?

Below is an example of the Engine command:

```
<eng order="1" setcmd="AntennaHopping,1"/>
```

- This command <eng/> tells us that it is the first command to run and it is a set command (setcmd) to the command interface.
- The pre-defined commands (cmd) consist of the following:
  - Start
  - Run – Equates to a delay between commands. The run command must have a duration attribute.
  - Stop
- The goto command allows us to return to a specific order number.
  - The goto command must refer to a valid <eng/> order attribute of the script.

```
<eng order="10" goto="2"/>
```


- The key command allows the script to be loaded from a file located in the \Scripts subdirectory of the TAVIS™ Suite Concentrator directory.

```
<eng order="11" key="script1"/>
```

- XML files and streams must begin with <tavsdk> where devicename is a device on the Concentrator, and include at least one <eng/> engine command, as follows:

```
<tavsdk name="devicename">
  <eng order="1" />
  ...
</tavsdk>
```

- Below is a full-script example:

Full-Script Example

```
<tavsdk name="devicename">
  <eng order="1" setcmd="AntennaHopping,1" />
  <eng order="2" setcmd="Antenna1On,1" />
  <eng order="3" setcmd="Antenna2On,1" />
  <eng order="4" setcmd="Antenna3On,0" />
  <eng order="5" setcmd="Antenna4On,0" />
  <eng order="6" setcmd="EPC1,1" />
  <eng order="7" setcmd="RFON,1" />
  <eng order="8" cmd="Start" />
  <eng order="9" cmd="Run" duration="5" />
  <eng order="10" cmd="Stop" />
  <eng order="11" goto="1" />
</tavsdk>
```


Script Engine Control Commands

The script engine also contains control commands to address specific functionality in the engine. The following functionality – (1) Control Alert notification, (2) stopping a running script, and (3) retrieving a list of all active scripts – can be performed using XML via the Msg command structure as described below:

```
<Msg MsgType="Command" CommandName="YYY">
  <YYY>XXX</YYY>
</Msg>
```

Where the valid CommandNames (YYY) are:

- Alert – Turn on alert notification for engine commands.
  - Valid XXX values are: 0, 1, Off, On.
- Active – Retrieves a list of active scripts or a specific script.
  - Valid XXX values are: All or a key name.
- Stop – Stop an active script.
  - Valid XXX value is a key name.

Syntax	`<Msg MsgType="Command" CommandName="YYY">`
	`  <YYY>XXX</YYY>`
	`</Msg>`

Response	`<Msg MsgType="Response" CommandName="Active">`
	`  <Table>`
	`  <ER>device1_test1_127.0.0.1_5876</ER>`
	`  <ER>device1_test1_127.0.0.1_6044</ER>`
	`  <ER>device1_test1_127.0.0.1_5476</ER>`
	`  </Table>`
	`</Msg>`

Example	`<Msg MsgType="Command" CommandName="Active">`
	`  <Active>All/Active>`
	`</Msg>`

8 – Component Identification

What is the Component Identification?

The Component Identification process uses UDP to detect and identify RF Code components on the local network segment. By default, the TAVIS™ Concentrator listens on port 60002 for UDP broadcast commands. This port is also used to send responses.

How will the identification message be sent?

The Identification Message consists of an Identify Command Query and Response. The following is an example of such a dialog in which the Service requests identification of all known Concentrator Components (ServiceName). In this case, a response is seen from a Component named “MyConcentratorName”.

Query

```
<Msg
    MsgType="Query"
    CommandName="Identify"
    ServiceName="Concentrator"
/>
```

Response

```
<Msg MsgType="Response">
  <Component
    Name=”MyConcentratorName”
    Func="Concentrator"
    Type="Service"
    Ver="2.2.0.01"
    ParentName="ControlStation"
    Listen="59001"
  />
</Msg>
```

A – Device Type Lookup Table

Device Friendly Name	Device Driver Name (Windows DLL)	Device Driver Name (Linux SO)
Mantis	mantis-N0010001.dll	libmantis.so
Mantis II	mantis2-N0010001.dll	libmantisII.so
Mantis II LT	mantis2LT-N0010001.dll	libmantisIILT.so
Spider	spider-N0010001.dll	libspider.so
Spider V	spiderv-N0010001.dll	libspiderv.so
Alien ALR	alien-alr-N0010001.dll	libalien-alr.so
Alien ALR-977X	AlienDriver_ALR977x.dll	libAlienDriver_ALR977x.so
Matrics AR400	matrics-ar400-N0010001.dll	libmatrics-ar400.so
AWID	awid-N0010001.dll	libawid.so
SAMSys9320V20	samsys-N0010002.dll	libsamsys-mp9320V20.so
SAMSys9320V27	samsys-N0020002.dll	libsamsys-mp9320V27.so
TIMicro	tirismicro-N0010001.dll	libtirismicro.so

Note – The XML device driver name is identical to the **Device Friendly Name** plus the XML file extension.

Note – The TAVIS™ suite is not limited to these drivers. The drivers are supplied with the current version of the TAVIS™ Concentrator 2.2.

B – Reserved Device Names

Reserved Name	Description
System	Indicates the system component (e.g. Concentrator)
Devices	Indicates all device components
All	Indicates all system and device components
None	Reserved for command syntax usage

C – Alert Event State Table

State	When Alert Occurs
UP	Component start
DOWN	Component stop
PRP	Peripheral condition (e.g. GROUPCODE mismatch)
CONNECTING	Start of connect attempt
RECONNECTING	Restart connect attempt
DEAD	Final fail to connect
STARTED	Process
STARTING	Process
STOPPING	Process
STOPPED	Process
ADD	Device or IP List
DELETE	Device or IP List
BLOCK	IP List
UNBLOCK	IP List

D – Component Status Table

Status	Value	Description
Stopped	1	Component has stopped operation
Stopping	2	Component is stopping operation
Starting	3	Component is starting operation
Started	4	Component has started operation
Error	5	Component has caused an error

E – System Properties

The Concentrator operations are bound to the System Property settings. These properties can be viewed or updated via the Get and Set commands respectively. Below is a list of RF Code Concentrator System Properties.

System Properties

* Read Only. ** Write Only. *** Changes take effect when the Concentrator is restarted.

Property Name	Data Type	Description
SystemName *	String	This is the system name.
MaxTime	Integer	This is the maximum time in seconds that a data buffer is to be released. Default: 10
MaxNumRecord	Integer	This is the maximum number of records (upper limit is 500,000) in a data buffer before the data buffer is released. Default: 10000
MaxAutoReconnects	Integer	A deprecated property, this is the maximum number of automatic attempts to connect to a device once a connection had been made. Default: 30 Note – This sets the MaxAutoReconnects device property of all devices.
ReconnectInterval	Integer	A deprecated property, this specifies the time interval in seconds between reconnection attempts. Default: 10 Note – This sets the ReconnectInterval device property of all devices.
DBOutput	Boolean	If set to 1, then the buffering to memory is enabled. If set to 0, then the buffering is disabled. Default: 0 (Off)
ConPortDataServer ***	Integer	This is the TCP port of the Data Server for the Concentrator Service (Direct Stream) (see page 21). This value is set in a configuration file and is read-only at runtime. Default: 6700
ConPortServer ***	Integer	This is the TCP port of the Command Interface where the commands are sent to control the Concentrator (see page 13). This value is set in a configuration file and is read-only at runtime. Default: 59001
ConPortClient ***	Integer	This specifies the port for UDP queries. Default: 60002
ConPortClientCol	Integer	This is not used in this version. Default: 59004
ConPortServBuff ***	Integer	This is the TCP port of the Command Interface (see page 13). This value is set in a configuration file and is read-only at runtime. Default: 51999
ConPortClientCS	String	This is not used in this version. Default: 59004

Property Name	Data Type	Description
AlertServerIP ***	String	This specifies the IP address of the Alert Agent. **Default:** 127.0.0.1
AlertServerPort ***	Integer	This specifies the port of the Alert Agent. **Default:** 7776
WatchDogTimer	Integer	A deprecated property, this specifies the time interval in seconds to wait before it starts reconnection attempts. **Default:** 2
Version *	String	This is the release version of the Concentrator application.
BufferIntegrity	Boolean	This sets the delete mode (auto/manual) for files created in the Concentrator buffering system. If set to 0, then the delete mode is Auto Delete. If set to 1, then the delete mode is Manual Delete, where only the **DeleteBuffer** command can delete files. **Default:** 0 (Auto)
DSMGRPort	Integer	This is the port used by the Driver System Manager (DSMGR) to receive commands.
DeviceTypes *	String	This lists the available device types in the Concentrator system.
SystemExec	N/A	For RF Code internal use only.
MonitorCurrent	N/A	For RF Code internal use only.

F – Device Properties

Note – Not all properties are supported by all devices. For more details on supported values, see the Hardware Vendor Specification and the RF Code Device Drivers user manual.

Common Driver Properties

* Read Only. ** Write Only. *** Changes take effect when the Concentrator is restarted.

Property Name	Data Type	Description
DeviceName*	String	Device Name. This is the distinctive name of the device (unchangeable unless the device is deleted and re-added).
DeviceType	String	Device Type. This property is set by the device driver upon setting of the DeviceDriver property. This property is not updated until a connection is made to the actual device. Note – This property accepts a value (R/W), however the value is overwritten with the actual device type by the device driver.
DeviceVersion*	String	Device Version. This is the firmware version of the device, as set by the device driver. This property is not updated until a connection is made to the actual device.
DeviceDriver	String	Device Driver. Device Friendly Name (see page 48, Appendix A).
DriverVersion*	String	Driver Version. This is the software version of the driver.
ConnectionMode	String	Connection Mode. This is the means to connect to a device, and can be set to IP address (network), serial port, etc. Default: EMPTY
ServerPort	Integer	Port Number. This is the TCP port on which the device listens for direct connections. Default: 6500
BaudRate	String List	Baud Rate. If ConnectionMode is serial, then this is the baud rate for the serial driver. Default: 38400
MaxAutoReconnects	Integer	Reconnection Attempts. If an error is detected, this specifies how many reconnects (0 to 100) the driver attempts. Default: 30
ReconnectInterval	Integer	Reconnection Interval (sec). This specifies the delay in seconds between reconnection attempts. Default: 10 (sec)
WatchDogTime	Integer	Watchdog Time Interval (sec). This is the time in seconds to delay the starting of reconnection attempts. Default: 2 Note – Some devices (readers) feature a built-in Fail Safe mechanism that reacts to line inactivity (no tags or commands) after a specified period. If status messages become unavailable (for an internally determined period), the driver delays WatchDogTime seconds before reconnection attempts are made.

Property Name	Data Type	Description
CheckStatusInterval	Integer	Check Status Interval (sec). This specifies the interval in seconds, for how often the device status is queried. The recommended setting is 4 seconds. If the driver does not get status information after MaxStatusFail retries, the driver will try to reconnect. Default: 5
MaxStatusFail	Integer	Maximum Status Failures Allowed. This is the maximum allowable number of status query failures. Default: 5
StreamOut	String List	Direct Stream Output. This is the flag that controls the device's Direct Stream output. If set to On (1), the device's data will appear in the Direct Stream. If set to Off (0), this will not occur. Default: On (1)
DeviceMode	String List	Device Mode. This is the device tag-reporting operation mode. Default: 20 Possible values are:
		MODE: Description
		20: Continuous mode. All tags as they come.
		E20: Exception mode. Tags are reported as New on the first beacon in a range. If a tag does not beacon during the TagTimeout interval, it is reported as Lost.
		31: Hitachi Tags.
		40: Japanese Tags.
		E40: Exception Mode for Japanese Tags.
		433: Continuous mode. The frequency is 433 MHz for a specific type of RF Code reader.
		E433: Exception mode. The frequency is 433 MHz for a specific type of RF Code reader.
TagTimeout	Integer	Tag Time Out Interval (sec). In Exception mode (E20, E40), this specifies the interval after a tag is first seen (New). If the tag does not beacon during this interval, it is reported as Lost. Default: 60

Time Stamp Format Parameter

Note – This time stamp format is supported only in the RF Code Mantis™, Mantis™-II and Spider™ V devices.

Roll Over	Decimal Places			
	0 Places (9)	1 Place (.9)	2 Places (.99)	3 Places (.999)
9 → 0	16	17	18	19
99 → 0	32	33	34	35
999 → 0	48	49	50	51
9999 → 0	64	65	66	67
99999 → 0	80	81	82	83
999999 → 0	96	97	98	99
9999999 → 0	112	113	114	115
99999999 → 0	128	129	130	131
999999999 → 0	144	145	146	147
9999999999 → 0	160	161	162	163

- Example – If the parameter equals 82, then the roll over will occur after 99999, and with 2 decimal places. Therefore, the time stamp display will range from 0.01 to 99999.99 seconds.

G – Messages Described

The following sections describe the various messages – alerts, responses, and errors – and their assignments.

Component ID Assignments

Combining a unique **Product ID** (up to three digits) and **Module ID** (two digits) forms a **Component ID** (up to five digits). The **Module ID** is unique within the **Product ID**.

Note – If the **Component ID** has leading zeros, they are eliminated. The following tables will denote the eliminated leading zeros with brackets [00].

Product ID	Product Name
[00]1	Concentrator
[00]2	Collector
[00]3	Control Station
101	Locator
102	Asset Tracking

Module ID	Module Name
01	Concentrator Service
02	Command Interface
03	Alert Agent
04	Search Agent
06	Driver System Manager (DSMGR)
07	Synchronization Service
20	Concentrator LI (GUI)
21	Alert Viewer
01	Collection Service
01	Control Station (GUI)
01	Locator (GUI)
01	Asset Tracking (GUI)

Comp. ID	Product ID	Module ID	Product Name	Module Name
[00000]	000	00	[General]	[General]
[00]101	001	01	Concentrator	Concentrator Service
[00]102	001	02	Concentrator	Command Interface
[00]103	001	03	Concentrator	Alert Agent
[00]104	001	04	Concentrator	Search Agent
[00]106	001	06	Concentrator	Driver System Manager (DSMGR)
[00]107	001	07	Concentrator	Synchronization Service
[00]120	001	20	Concentrator	Concentrator LI (GUI)
[00]121	001	21	Concentrator	Alert Viewer
[00]201	002	01	Collector	Collection Service
[00]301	003	01	Control Station	Control Station (GUI)
10101	101	01	Locator	Locator (GUI)
10201	102	01	Asset Tracking	Asset Tracking (GUI)

- Example – The Component ID = [00]102 refers to the Concentrator and Command Interface.

Message Number Assignments

A Message Number is a unique number (up to ten digits) formed by combining a Component ID (up to five digits) and a Message ID (up to five digits).

- However, if the Message Number has leading zeros, they are eliminated. The following tables will denote the eliminated leading zeros with brackets (e.g. [00], [00000]).

Example – The Message Number = [00]10111020 refers to the "Purge started" message response from the Command Interface, where:

- Component ID = [00]101 (Concentrator-Concentrator Service).
- Message ID = 11020 (Purge Started).
- However, because there are two leading zeros, the Message Number will appear as an eight-digit number as follows.
- Message Number = [00]10111020 (the first two leading zeros are eliminated).

General Messages

These messages are general or common to all components, and as such, use the common Component ID = [00000].

Example – The Message Number = [0000000000] refers to the "OK" message response from any component, where:

- Component ID = [00000] (GENERAL-Common)
- Message ID = [00000] (OK)

Component ID = [00000]

- [00000] - General
- Message Number = [00000] + Message ID (see note below)

Component ID = [00]101

- [00]101 – Concentrator – Concentrator Service
- Message Number = [00]101 + Message ID (see note below)

Component ID = [00]102

- [00]102 – Concentrator – Command Interface
- Message Number = [00]102 + Message ID (see note below)

Component ID = [00]103

- [00]103 – Concentrator – Alert Agent
- Message Number = [00]103 + Message ID (see note below)

Component ID = [00]104

- [00]104 – Concentrator – Search Agent
- Message Number = [00]104 + Message ID (see note below)

Component ID = [00]106

- [00]106 – Concentrator – Driver System Manager (DSMGR)
- Message Number = [00]106 + Message ID (see note below)

Note – For the complete list of Message IDs, open the tavMsgIds.xml file located in the \SYSTEM32\TAVIS\config\ folder of the Windows directory. In Linux, this file is located in the \opt\TAVIS\config\ directory.

H – Glossary

Active Tag – This is an RFID tag that comes with a battery that is used to power the microchip's circuitry and transmit a signal to a reader. Active tags can be read from 100 feet (30.5 meters) or more away. They can be used for tracking items over long ranges. For instance, the US military uses active tags to track containers of supplies arriving in ports.

Amplitude – This is the maximum absolute value of a periodic curve measured along its vertical axis (in non-technical language, the height of a wave).

Antenna – The antenna is the conductive element that enables the tag to send and receive data. A passive tag usually has a coiled antenna that couples with the coiled antenna of the reader to form a magnetic field. A passive tag draws power from this field.

Anti-Collision – This is a general term used to cover methods of preventing radio waves from one device from interfering with radio waves from another. Anti-collision algorithms are also used to read more than one tag in the same reader's field.

Auto-ID – See Automatic Identification.

Automatic Identification (Auto-ID) – Sometimes called "automatic data capture," these are methods of collecting data and entering it directly into computer systems without human involvement. Technologies normally considered part of Auto-ID include bar codes, biometrics, RFID and voice recognition.

Back Scatter – This is a method of communication between tags and readers. RFID tags using back-scatter technology reflect back to the reader a portion of the radio waves that reach them. The reflected signal is modulated to transmit data. Tags using back-scatter technology can either be passive or active, but in each case, they are more expensive than tags that use inductive coupling.

Bar Code – This is a standard method of identifying the manufacturer and product category of a particular item. The barcode was adopted in the 1970s because the bars were easier for machines to read than optical characters. However, the main drawbacks of barcodes are they do not identify unique items and scanners have to have a line-of-sight to read them.

Contactless Smart Card – This refers to a credit card or loyalty card that contains an RFID chip, in order to transmit information to a reader without having to be swiped through a reader. Such cards can speed checkout, providing consumers with more convenience.

Chipless RFID Tag – This is an RFID tag that does not depend on an integrated microchip. Instead, the tag uses materials that reflect back a portion of the radio waves. A computer takes a snapshot of the waves beamed back and uses it (like a fingerprint)to identify the object with the tag.

Closed-Loop Systems – These are RFID tracking systems set up within a company. Since the tracked item never leaves the company's control, using technology based on open standards is not a priority requirement.

Coupling – See Inductive Coupling.

Electromagnetic Compatibility (EMC) – This is the ability of a system or product to function properly in an environment where other electromagnetic devices are used, and not be a source itself of electromagnetic interference.

Electromagnetic Interference (EMI) – This is the interference caused when the radio waves of one device distort the waves of another. Cell phones, wireless computers and even robots in factories can produce radio waves that interfere with RFID tags.

Electronic Article Surveillance (EAS) – These are electronic tags that can be turned on or off. When an item is purchased (or borrowed from a library), the tag is turned off. When someone passes a gate area holding an item with a tag that has not been turned off, an alarm sounds. EAS tags are embedded in the packaging of most pharmaceuticals.

Electronic Product Code (EPC) – This is the 96-bit code that may one day replace barcodes. The EPC has digits to identify the manufacturer, product category and the individual item. It is backed by the United Code Council and EAN International, the two main bodies that oversee barcode standards.

Error-Correcting Code – This is the code stored on an RFID tag to enable the reader to figure out the value of missing or garbled bits of data. This is needed for those cases when a reader may misinterpret some data from the tag (e.g. a Rolex watch is misinterpreted as a pair of socks).

Error-Correcting Mode – This is the mode of data transmission between the tag and reader in which errors or missing data is automatically corrected.

Error-Correcting Protocol – This is the set of rules used by readers to interpret data correctly from the tag.

European Article Numbering (EAN) – This is the barcode standard used throughout Europe, Asia and South America. It is administered by EAN International.

Excite – The reader is said to "excite" a passive tag when the reader transmits RF energy to wake up the tag and enable it to transmit back.

eXtensible Markup Language (XML) – This is a widely accepted way of sharing information over the Internet in a way that computers can use, regardless of their operating system.

Factory Programming – Some read-only tags must have their identification number written into the silicon microchip at the time the chip is made. The process of writing the ID number into the chip is called “factory programming.”

Field Programming – Tags that use EEPROM, or non-volatile memory, can be programmed after it is shipped from the factory. This feature is called “field programming.”

Frequency – This is the number of repetitions of a complete wave within one second, where 1 Hz equals one complete waveform in one second, and 1 KHz equals 1,000 waves in a second. RFID tags use low, high, ultra-high and microwave frequencies. Each frequency has advantages and disadvantages that make it more suitable for some applications than for others.

Global Positioning System (GPS) – This is a system that is used to locate an exact position on the Earth anytime, anywhere. GPS satellites, at least 24 in all, orbit at 11,000 nautical miles above the Earth. They are continuously monitored by ground stations located worldwide. The satellites transmit signals that can be detected by anyone with a GPS receiver. Using the receiver, a location can be determined with great precision.

Global Tag (GTAG) – This tag is a standardization initiative of the Uniform Code Council (UCC) and the European Article Numbering Association (EAN) for asset tracking and logistics based on radio frequency identification (RFID).

High-Frequency Tags – These tags typically operate at 13.56 MHz. They can be read from about 10 feet away and transmit data faster. However, in doing so, they consume more power than low-frequency tags.

Inductive Coupling – This is a method of transmitting data between tags and readers in which the antenna from the reader picks up changes in the tag's antenna.

Industrial, Scientific, and Medical (ISM) Bands – These are a group of unlicensed frequencies in the electromagnetic spectrum.

Integrated Circuit (IC) – This is a microelectronic semiconductor device composed of many interconnected transistors and other components. Most RFID tags have ICs.

Interrogator – See Reader.

Low-Frequency Tags – These tags typically operate at 125 KHz. The main disadvantages of low-frequency tags are that they have to be read from within 3 feet (0.9 meters) and the rate of data transfer is slow. However, they are less expensive and less prone to interference than high-frequency tags.

Microwave Tags – These are radio frequency tags that operate at 5.8 GHz. They have very high transfer rates and can be read from as far as 30 feet (9.1 meters) away, but they consume a lot of power and are expensive.

Modulation – This refers to changing the frequency or amplitude of a wave, in order to transmit data that is converted into digital form. For example, a wave with the normal amplitude (or height) may be a "1" in binary code, while a wave with a lower amplitude might be a "0".

Multiple Access Schemes – These are methods of increasing the amount of data that can be transmitted wirelessly within the same frequency spectrum. For example, RFID readers use Time Division Multiple Access (or TDMA) to read tags at different times to avoid interfering with one another.

Multiplexer – This is an electronic device that allows a reader to have more than one antenna. Each antenna scans the field in a preset order.

Nominal Range – This is the read range at which the tag can be read reliably.

Null Spot – This is the area in the reader field that does not receive radio waves. Essentially, this is the reader's blind spot. It is a phenomenon common to UHF systems.

Object Name Service (ONS) – This is an Auto-ID Center-designed system for looking up unique Electronic Product Codes (EPCs) and pointing computers to information about the item associated with the code. ONS is similar to the Domain Name Service (DNS), which points computers to sites on the Internet.

Passive Tag – This is an RFID tag without a battery. When radio waves from the reader reach the chip's antenna, it creates a magnetic field. The tag draws power from the field and is able to send back information stored on the chip. Today, simple passive tags cost around 0.5 to several US dollars.

Patch Antenna – This is a small square antenna made from a solid piece of metal or foil.

Physical Markup Language (PML) – This is an Auto-ID Center-designed method of describing products in a way computers can understand. PML is based on the widely accepted eXtensible Markup Language (XML) used to share data over the Internet in a format all computers can use, regardless of their operating system.

PML Server – This is a server that responds to requests for Physical Markup Language (PML) files related to individual Electronic Product Codes (EPCs). The PML files and servers will be maintained by the manufacturer of the item.

Power Level – This is the amount of RF energy radiated from a reader or an active tag. The higher the power output, the longer the read range, but most governments regulate power levels to avoid interference with other devices.

Programming – This refers to writing data to an RFID tag.

Proximity Sensor – This is a device that detects the presence of an object and signals another device. Proximity sensors are often used on manufacturing lines to alert robots or routing devices on a conveyor to the presence of an object.

Radio Frequency Identification (RFID) – This is a method of identifying unique items using radio waves. RFID hardware currently consists of two products: Tags and Readers. Typically, a reader communicates with a tag, which holds digital information in a microchip. However, there are chipless forms of RFID tags that use material to reflect back a portion of the radio waves beamed at them.

- Tags – RF Code tags are small and inexpensive electronic "bar code". A tag is usually a chip attached to a tiny antenna and housed in a plastic case. Originally, the chip's responsibility was simply to transmit a unique number. Today, tags contain memory and can retain a large amount of information about the product to which they are attached. Tags also can be programmed to have a life of their own. That is, they can announce their location when they have been moved, and provide information, such as location, security, presence, temperature or pressure, from devices to which they are attached.
- Readers – Readers are used to identify the location of assets in virtually real time. Readers can be as simple as a loop around a doorway to record tagged items arriving at or leaving the premises. Tags can also be attached to people, such as a child when attending an amusement part or zoo. When attached to an asset, a tag, upon motion or upon tampering, transmits at predefined intervals a unique identification code to a network of transceivers, or readers. These readers pass the message transmissions to the server software for decoding. The asset's movement is then graphically displayed in virtually real time and stored.

Range – See Read Range.

Read – The process of turning radio waves from a tag into bits of information that can be used by computer systems.

Read Rate – The maximum rate at which data can be read from a tag expressed in bits or bytes per second.

Reader – The reader (also called an interrogator) communicates with the RFID tag via radio waves and passes the information in digital form to a computer system.

Reader Field – The area of coverage. Tags outside the reader field do not receive radio waves and can't be read.

Read-Only Tags – Tags that contain data that cannot be changed unless the microchip is reprogrammed electronically.

Read Range – The distance from which a reader can communicate with a tag. Active tags have a longer read range than passive tags because they use a battery to transmit signals to the reader. With passive tags, the read range is influenced by frequency, reader output power, antenna design, and method of powering up the tag. Low frequency tags use inductive coupling (see above), which requires the tag to be within a few feet of the reader.

Read-Write Tags – RFID tags that can store new information on its microchip. San Francisco International Airport uses a read-write tag for security. When a bag is scanned for explosives, the information on the tag is changed to indicate it has been checked. The tag is scanned again before it is loaded on a plane. Read-write tags are more expensive than read only tags, and therefore are of limited use for supply chain tracking.

RFID Tag – A microchip attached to an antenna that picks up signals from and sends signals to a reader. The tag contains a unique serial number, but may have other information, such as a customers' account number. Tags come in many forms, such smart labels that are stuck on boxes; smart cards and key-chain wands for paying for things; and a box that you stick on your windshield to enable you to pay tolls without stopping. RFID tags can be active tags, passive tags and semi-passive tags.

Real Time Location System (RTLS) – RTLS is RFID that is upgraded to produce instant location information.

- Items tagged with normal RFID are read when a reader is brought near a tag or a tagged item is passed near a reader. RTLS allows a reader unit to "see" the actual location of a tagged item, without the tagged item being near the reader. Using special readers placed around a property, tags are located using a triangulation system. Tagged items may be as small as a tool or as large as a trailer truck.
- RTLS readers are usually deployed as a matrix of readers that are installed at a spacing of anywhere from 50 to 1000 feet. The system continually updates the database with current tag locations as frequently as every several seconds, or as infrequently as every few hours for items that seldom move. Most RTLS tag transmissions come from a combination of timed intervals, tag movement, low battery warning, and response to inquirers from the reader system. The frequency of tag location updates may have implications for the number of tags that can be deployed and the battery life of the tag. A typical applications system can track thousands of tags simultaneously and the average tag battery life can be three to four years.

Scanner – An electronic device that can send and receive radio waves. When combined with a digital signal processor that turns the waves into bits of information, the scanner is called a reader or interrogator.

Semi-Passive Tag – Similar to active tags, but the battery is used to run the microchip's circuitry but not to communicate with the reader. Some semi-passive tags sleep until they are woken up by a signal from the reader, which conserves battery life. Semi-passive tags cost a dollar or more.

Sensor – A device that responds to a physical stimulus and produces an electronic signal. Sensors are increasingly being combined with RFID tags to detect the presence of a stimulus at an identifiable location.

Silent Commerce – This term covers all business solutions enabled by tagging, tracking, sensing and other technologies, including RFID, which make everyday objects intelligent and interactive. When combined with continuous and pervasive Internet connectivity, they form a new infrastructure that enables companies to collect data and deliver services without human interaction.

Smart Label – A label that contains an RFID tag. It's considered "smart" because it can store information, such as a unique serial number, and communicate with a reader.

Smart Card – See Contactless Smart Card.

Tag – See RFID Tag.

TAVIS™ – A software system developed by RF Code to provide a Total Asset VISibility (TAVIS) system.

Time Division Multiple Access (TDMA) – A method of solving the problem of the signals of two readers colliding. Algorithms are used to make sure the readers attempt to read tags at different times.

Transponder – A radio transmitter-receiver that is activated when it receives a predetermined signal. RFID tags are sometimes referred to as transponders.

Ultra-High Frequency (UHF) – Typically, tags that operate between 866 MHz to 930 MHz. They can send information faster and farther than high- and low-frequency tags. However, radio waves don't pass through items with high water content, such as fruit, at these frequencies. UHF tags are also more expensive than low-frequency tags, and they use more power.

Uniform Code Council (UCC) – The nonprofit organization that oversees the Uniform Product Code, the barcode standard used in North America.

Uniform Product Code (UPC) – The barcode standard used in North America. It is administered by the Uniform Code Council.

Write Rate – The rate at which information is transferred to a tag, written into the tag's memory and verified as being correct.

XML – See eXtensible Markup Language (XML).

XML Query Language (XQL) – A method of querying a database based on XML.

I – Index

J – Copyright Statement

TAVIS™ Concentrator 2.2 – SDK Suite
User Manual
Revision 1.01

Written and designed by RF Code. Technical Writer: Bob Litt.

RF Code, Inc.
1250 South Clearview Avenue
Mesa, AZ 85208 USA

www.rfcode.com

Technical Support
Visit our website. Click the “Contact Us” link.