APC
by Schneider Electric

BotzWare 3.x Web Services Specification

Table of Contents

1 Web Services Overview

The NetBotz BotzWare Web Services interface provides a mechanism that the NetBotz appliances may utilize for delivering alerts. This mechanism is in the form of third-party Web Services that implement a provided interface standard.

The samples and documentation provided for the NetBotz Web Services tend to favor the Microsoft Visual Studio .NET platform, due to the fact that this is the most popular web services development platform at present. Despite this, every effort has been made to assure that the NetBotz Web Services implementation will be fully interoperable with other web services frameworks. For other platforms, the WSDL for the various interfaces is provided and should be used in combination with the documentation here to allow use by non-.NET platforms. Several samples implemented with the gSoap library on Linux are also included.

The library used for the implementation of the NetBotz Web Services within the NetBotz appliances is gSoap V2.7.0f. To assure maximum interoperability, the WSDL used for the implementation is derived from a Visual Studio .NET reference implementation (coded in C#) and then used to generate the interfaces implemented with the gSOAP library. The gSOAP library claims interoperability with many SOAP and Web Service frameworks, including:

- Apache 2.2
- Apache Axis
- ASP.NET
- Cape Connect
- Delphi
- easySOAP++
- eSOAP
- Frontier
- GLUE
- Iona XMLBus
- kSOAP
- MS SOAP
- Phalanx
- SIM
- SOAP::Lite
- SOAP4R
- Spray
- SQLData

- Wasp Adv.
- Wasp C++
- White Mesa
- xSOAP
- ZSI
- 4S4C

2 Alert Receiver Web Service API

2.1 Overview

In order to allow quick and efficient response to alert conditions, a special web services specific alert action is available as of BotzWare V2.2.2. Rather than providing a "call in" web service, the "Call Web Service Alert Receiver" alert action implements a web services client that can be directed to "call" a web service at an arbitrary URL. As the alert action needs to have prior knowledge of the binding and interface for the web service (i.e., what methods are implemented, parameter and output formats, etc), the interface to be implemented by the web service is fixed and defined by NetBotz.

The interface for an Alert Receiver web service implementation currently consists of five method calls, each of which are required to be implemented (even if they do not provide any function). For any given alert, the "Call Web Service Alert Receiver" action will call exactly one of these interfaces, depending on what type of alert is being processed. In all cases, the method is invoked with parameters containing objects describing the appliance, the specific alert, and the object the alert is associated with (sensor, port, pod). Whether the alert is processed or not, the web service implementation should quickly process the call and return "true" if processing was completed successfully or "false" if processing failed. If "false" is returned, the alert action will both log the error and attempt to call any configured "backup" service (i.e., fail-over).

The alert action supports use of SSL, including supplying the appliance's certificate for client authentication and optionally verifying server certificates. User-ID/Password authentication is also supported.

Implementations of the Alert Receiver Web Service interface must match the interface described in the "WSDL for Alert Receiver Web Services" document, with the sole exception of the URL specified in the <soap:address> tag under the <wsdl:port> for the "nbAlertReceiverServiceSoap". The alert action will assume that the URL provided when the alert action was configured is the correct value for this field.

2.2 Interface Definitions

2.2.1 BotzWareNumericSensorAlert() – Alert Web Service for Numeric Sensor Alerts

C# Binding	Public bool BotzWareNumericSensorAlert(nbPod Appliance, nbAlert Alert, nbNumSensor Sensor, nbPort PortForSensor, nbPod PodForSensor, bool IsResolved, string UserArg, nbUnitsRequest Units);
VB.NET Binding	Public Function BotzWareNumericSensorAlert (Appliance as nbPod, Alert as nbAlert, Sensor as nbNumSensor, PortForSensor as nbPort, PodForSensor as nbPod, IsResolved as Boolean, UserArg as String, Units as nbUnitsRequest) as Boolean
C++.NET Binding	bool BotzWareNumericSensorAlert(nbPod __gc *Appliance, nbAlert __gc *Alert, nbNumSensor __gc *Sensor, nbPort __gc *PortForSensor, nbPod __gc *PodForSensor, bool IsResolved, String __gc *UserArg, enum nbUnitsRequest Units);
SOAP Action	http://www.netbotz.com/BotzWare/BotzWareNumericSensorAlert
Parameters	Appliance – This is the nbPod object for the appliance (i.e., the "nbBaseEnclosure" object). This includes the key information for identifying the device sending the alert. See nbPod class for details. Alert – This is the nbAlert object describing the alert being processed. It includes the key data on the cause of the alert and its current status. See nbAlert class for details. Sensor – This is the nbNumSensor object describing the numeric sensor that the alert is associated with, including its current value and status. See nbNumSensor class for details. PortForSensor – This is the nbPort, if any, associated with the alerting sensor. Value is null/NULL/Nothing if there is no port. See nbPort for details. PodForSensor – This is the nbPod associated with the alerting sensor. See nbPod for details.

	IsResolved – This Boolean is set to “false” if the alert is active and “true” if the alert has been resolved (returned to normal). UserArg – This string is a free-form, user-definable string that can be provided when the alert action is configured. It can include resolved macros or any other text the user defines. Units – This nbUnitsRequest value indicates the units system option selected by the user when configuring the alert action. This setting also drives the units system used in the data contained within the *Sensor* object.
Outputs	Returns “true” if the alert is successfully received and processed, “false” if the processing failed.
Privilege Required	N/A
BotzWare Version	V2.2.2 and later
Description	This method is invoked by the “Call Web Service Alert Receiver” alert action when the alert being processed is associated with a numeric sensor (i.e. a threshold alert for a numeric sensor, a sensor unplugged condition, etc).

2.2.2 BotzWareStateSensorAlert() – Alert Web Service for State Sensor Alerts

C# Binding	Public bool BotzWareStateSensorAlert (nbPod Appliance, nbAlert Alert, nbStateSensor Sensor, nbPort PortForSensor, nbPod PodForSensor, bool IsResolved, string UserArg);
VB.NET Binding	Public Function BotzWareStateSensorAlert (Appliance as nbPod, Alert as nbAlert, Sensor as nbStateSensor, PortForSensor as nbPort, PodForSensor as nbPod, IsResolved as Boolean, UserArg as String) as Boolean
C++.NET Binding	bool BotzWareStateSensorAlert(nbPod __gc *Appliance, nbAlert __gc *Alert, nbStateSensor __gc *Sensor, nbPort __gc *PortForSensor, nbPod __gc *PodForSensor, bool IsResolved, String __gc *UserArg);
SOAP Action	http://www.netbotz.com/BotzWare/BotzWareStateSensorAlert

Parameters	Appliance – This is the nbPod object for the appliance (i.e., the "nbBaseEnclosure" object). This includes the key information for identifying the device sending the alert. See nbPod class for details. Alert – This is the nbAlert object describing the alert being processed. It includes the key data on the cause of the alert and its current status. See nbAlert class for details. Sensor – This is the nbStateSensor object describing the state sensor that the alert is associated with, including its current value and status. See nbStateSensor class for details. PortForSensor – This is the nbPort, if any, associated with the alerting sensor. Value is null/NULL/Nothing if there is no port. See nbPort for details. PodForSensor – This is the nbPod associated with the alerting sensor. See nbPod for details. IsResolved – This Boolean is set to "false" if the alert is active and "true" if the alert has been resolved (returned to normal). UserArg – This string is a free-form, user-definable string that can be provided when the alert action is configured. It can include resolved macros or any other text the user defines.
Outputs	Returns "true" if the alert is successfully received and processed or "false" if the processing failed.
Privilege Required	N/A
BotzWare Version	V2.2.2 and later
Description	This method is invoked by the "Call Web Service Alert Receiver" alert action when the alert being processed is associated with a state sensor (i.e., a threshold alert for a state sensor, a sensor unplugged condition, etc).

2.2.3 BotzWareStringSensorAlert() – Alert Web Service for String Sensor Alerts

C# Binding	Public bool BotzWareStringSensorAlert (nbPod Appliance, nbAlert Alert, nbStringSensor Sensor, nbPort PortForSensor, nbPod PodForSensor, bool IsResolved, string UserArg);
VB.NET Binding	Public Function BotzWareStringSensorAlert (Appliance as nbPod, Alert as nbAlert, Sensor as nbStringSensor, PortForSensor as nbPort, PodForSensor as nbPod, IsResolved as Boolean, UserArg as String) as Boolean
C++.NET Binding	bool BotzWareStringSensorAlert (nbPod __gc *Appliance, nbAlert __gc *Alert, nbStringSensor __gc *Sensor, nbPort __gc *PortForSensor, nbPod __gc *PodForSensor, bool IsResolved, String __gc *UserArg);
SOAP Action	http://www.netbotz.com/BotzWare/BotzWareStringSensorAlert
Parameters	Appliance – This is the nbPod object for the appliance (i.e., the "nbBaseEnclosure" object). This includes the key information for identifying the device sending the alert. See nbPod class for details. Alert – This is the nbAlert object describing the alert being processed. It includes the key data on the cause of the alert and its current status. See nbAlert class for details. Sensor – This is the nbStringSensor object describing the string sensor that the alert is associated with, including its current value and status. See nbStringSensor class for details. PortForSensor – This is the nbPort, if any, associated with the alerting sensor. Value is null/NULL/Nothing if there is no port. See nbPort for details. PodForSensor – This is the nbPod associated with the alerting sensor. See nbPod for details. IsResolved – This Boolean is set to "false" if the alert is active and "true" if the alert has been resolved (returned to normal). UserArg – This string is a free-form, user-definable string that can be provided when the alert action is configured. It can

	include resolved macros or any other text the user defines.
Outputs	Returns “true” if alert successfully received and processed, “false” if the processing failed.
Privilege Required	N/A
BotzWare Version	V2.2.2 and later
Description	This method is invoked by the “Call Web Service Alert Receiver” alert action when the alert being processed is associated with a string sensor (i.e., a threshold alert for a string sensor, a sensor unplugged condition, etc).

2.2.4 BotzWarePortAlert() – Alert Web Service for Port Alerts

C# Binding	Public bool BotzWarePortAlert (nbPod Appliance, nbAlert Alert, nbPort Port, nbPod PodForPort, bool IsResolved, string UserArg);
VB.NET Binding	Public Function BotzWarePortAlert (Appliance as nbPod, Alert as nbAlert, Port as nbPort, PodForPort as nbPod, IsResolved as Boolean, UserArg as String) as Boolean
C++.NET Binding	bool BotzWarePortAlert (nbPod __gc *Appliance, nbAlert __gc *Alert, nbPort __gc *Port, nbPod __gc *PodForPort, bool IsResolved, String __gc *UserArg);
SOAP Action	http://www.netbotz.com/BotzWare/BotzWareStringSensorAlert
Parameters	Appliance – This is the nbPod object for the appliance (i.e., the “nbBaseEnclosure” object). This includes the key information for identifying the device sending the alert. See nbPod class for details. Alert – This is the nbAlert object describing the alert being processed. It includes the key data on the cause of the alert and its current status. See nbAlert class for details. Port – This is the nbPort object describing the port that the alert is associated with, including its current value and status. See nbPort class for details. PodForPort – This is the nbPod associated with the alerting port. See nbPod for details.

	IsResolved – This Boolean is set to “false” if the alert is active and “true” if the alert has been resolved (returned to normal). UserArg – This string is a free-form, user-definable string that can be provided when the alert action is configured. It can include resolved macros or any other text the user defines.
Outputs	Returns “true” if the alert is successfully received and processed or “false” if the processing failed.
Privilege Required	N/A
BotzWare Version	V2.2.2 and later
Description	This method is invoked by the “Call Web Service Alert Receiver” alert action when the alert being processed is associated with a port (but not a sensor).

2.2.5 BotzWarePodAlert() – Alert Web Service for Pod Alerts

C# Binding	Public bool BotzWarePodAlert (nbPod Appliance, nbAlert Alert, nbPod Pod, bool IsResolved, string UserArg);
VB.NET Binding	Public Function BotzWarePodAlert (Appliance as nbPod, Alert as nbAlert, Pod as nbPod, IsResolved as Boolean, UserArg as String) as Boolean
C++.NET Binding	bool BotzWarePodAlert (nbPod __gc *Appliance, nbAlert __gc *Alert, nbPod __gc *Pod, bool IsResolved, String __gc *UserArg);
SOAP Action	http://www.netbotz.com/BotzWare/BotzWarePodAlert
Parameters	Appliance – This is the nbPod object for the appliance (i.e., the “nbBaseEnclosure” object). This includes the key information for identifying the device sending the alert. See nbPod class for details. Alert – This is the nbAlert object describing the alert being processed. It includes the key data on the cause of the alert and its current status. See nbAlert class for details. Pod – This is the nbPod object describing the pod that the alert is associated with, including its current value and status. See

	nbPod class for details. IsResolved – This Boolean is set to “false” if the alert is active and “true” if the alert has been resolved (returned to normal). UserArg – This string is a free-form, user-definable string that can be provided when the alert action is configured. It can include resolved macros or any other text the user defines.
Outputs	Returns “true” if the alert is successfully received and processed or “false” if the processing failed.
Privilege Required	N/A
BotzWare Version	V2.2.2 and later
Description	This method is invoked by the “Call Web Service Alert Receiver” alert action when the alert being processed is associated with a pod (but not a sensor or port). In general, any non-specific alert (such as security alerts, network failures) are also delivered using this interface (as they are associated with the “nbBaseEnclosure” pod representing the appliance).

2.3 Samples and Skeletons for .NET

The "webservices" directory on the *NetBotz Appliance Utility CD* contains several sample projects for Microsoft Visual Studio .NET, which may be used to create alert receivers that conform to the interface requirements of the "Call Web Service Alert Receiver" alert action. Specifically, the following projects are provided:

- nbAlertReceiverSkeletonBasic – this is a skeleton interface, implemented in Visual Basic .NET. The project can be placed under the "wwwroot" of an IIS server running ASP.NET, registered, and modified freely.

- nbAlertReceiverSkeletonCsharp – this is a skeleton interface, implemented in Visual C# .NET. The project can be placed under the "wwwroot" of an IIS server running ASP.NET, registered, and modified freely.

- nbAlertReceiverSkeletonCPlusPlus – this is a skeleton interface, implemented in Visual C++.NET. The project can be placed under the "wwwroot" of an IIS server running ASP.NET, registered, and modified freely.

- nbSampleAlertReceiver – this is a simple application, written in Visual C#, that receives alerts, processes them, and generates simple summary files under the c:\test directory for each alert received. It offers a good demonstration of the sort of information available within the alert calls.

All the samples are implemented using the nbWebServiceAPI.dll, as this supplies the Com.NetBotz.BotzWare.DataClasses namespace needed for the parameters passed in the method calls.

Non-Visual Studio projects should use the WSDL for the Alert Receiver Service to define and produce their alert receiver implementations. The WSDL can be found in the webservices/wsdl/nbAlertReceiver.wsdl file on the *NetBotz Appliance Utility CD*.

3 Web Services Data Definitions

3.1 Overview

This section details the data classes and enumerations used for the parameters and return values of the various web services defined in this document. Like all the symbols defined here, these classes and interfaces are part of the "http://www.netbotz.com/BotzWare" namespace and are consistent across all the web service definitions covered here.

If the user is using the .NET platform for development, the nbWebServicesAPI.dll library for Microsoft .NET is **highly** recommended. The DLL provides a single consistent namespace and set of object definitions. Users and application developers are free to redistribute this DLL with their applications, subject to the NetBotz Redistribution License. Any of the .NET languages can use this DLL to access the NetBotz data objects by including the DLL as a "Reference" in their Visual Studio project. In the nbWebServicesAPI.dll, all data object definitions are provided under the Com.NetBotz.BotzWare.DataClasses namespace. All data classes are Common Language Specific (CLS) compliant, and should function correctly from all .NET languages.

All objects share a common theme. For example, pods, ports, sensors, alerts, etc. on a BotzWare-based appliance have an "object ID" (a string containing only alphanumeric characters and underscores), which is used to identify the object. The object ID for a given object is "locally unique" (i.e., distinct within the given appliance); although, some IDs will be common between different appliances (i.e., the object ID of the nbPod object for the appliance itself is always "nbBaseEnclosure"). Object IDs for a given object are unchangeable once the object is created.

While the definitions described below are in Visual Basic and Visual C# terms, the information should be logically consistent with any classes or structures derived from the WSDL for these interfaces. Applications needing to produce their own stubs (rather than having code automatically generated from the WSDL) should use the combination of WSDL for the interface definitions and the descriptions here to fully understand the interfaces.

The data types used in each class definition are either class references to other classes in the specification, enumerations defined in the specification, or standard .NET data types. For simple data types, the C# data type is used. The following table presents the Visual C++.NET and Visual Basic equivalents for these types.

Visual C# Type	**Visual C++.NET Type**	**Visual Basic Type**
double	double	Double
string	String __gc *	String
bool	bool	Boolean
int	int	Integer
null	NULL	Nothing
string[]	String __gc * __gc[]	String()
DateTime	DateTime __gc *	DateTime
byte[]	byte __gc * __gc[]	Byte()

3.2 Class Definitions

3.2.1 nbObject – abstract base class for all objects

Class: nbObject	Parent Class: n/a
Fields	
ID (string)	Object ID of instance
GUID (string)	Globally unique ID of instance (includes appliance serial number)
Class (string)	BotzWare class ID (may be different than class of web services object)
ClassPath (string[])	Array of BotzWare class IDs, ordered from base class to object's class
ReadOnly (bool)	Object is read-only
Constant (bool)	Value of object is a constant
NoDelete (bool)	Object cannot be deleted
Persistent (bool)	If true, object is persistent (will survive a reboot)
Description: This is the generic base class for all objects. Most of the information here, other than the *ID* and *GUID* fields, will not be of much use. Some objects, such as sensors, have a "deeper" class hierarchy with the BotzWare than is shown in the Web Services Data Classes – the *Class* and *ClassPath* fields can be useful for accessing this information.	

3.2.2 nbLocationData – location attributes for an object

Class: nbLocationData	Parent Class: n/a
Fields	
Location (string)	Generic location string
Country (string)	Country

State (string)	State, Province, or Territory
City (string)	City
Address1 (string)	First line of street address
Address2 (string)	Second line of street address
Company (string)	Company name
Site (string)	Site name
Building (string)	Building name
Floor (string)	Floor number
Room (string)	Room number
Latitude (string)	Latitude
Longitude (string)	Longitude
RoomRow (string)	Row within room
RoomCol (string)	Column within room
EnclosureID (string)	Rack or enclosure ID
SlotLocation (string)	Slot location within rack/enclosure
RelativeLocation (nbEnclosureRelLoc)	Relative position within enclosure (see nbEnclosureRelLoc enumeration)
HeightAboveFloor (string)	Height above floor
Contact (string)	Contact name or information
Notes (string)	Free-form notes on location
Description: This class is used to contain location attributes for pods, sensors, and other devices that support location information. The values for the various fields are inherited "down," so the City or Country of the appliance (the nbBaseEnclosure pod) is used for any other pods (and, through them, their sensors) unless those pods have differing values defined for those fields. Undefined field values will be null.	

3.2.3 nbProductData – product identification attributes for an object

Class: nbProductData	Parent Class: n/a
Fields	
Vendor (string)	Product vendor name
Type (string)	General type of product
Model (string)	Model of product
FullModel (string)	Fully qualified model name
SubModel (string)	Sub-model (i.e., ship group)
SerialNum (string)	Product serial number
Manufacturer (string)	Product manufacturer name
ManufDate (DateTime)	Manufacture date
Revision (string)	Product revision
BoardID (string)	Hardware board ID
BootVersion (string)	Boot code version
AppVersion (string)	Application code version
Description: This class is used to contain product information attributes for pods, applications, and other objects. Undefined field values will be null.	

3.2.4 nbPod – pod or appliance object

Class: nbPod	Parent Class: nbObject
Fields	
State (nbPodState)	Current state for pod (see nbPodState enumeration)
Label (string)	Label for pod
Parent (string)	Object ID of parent pod (null=no parent)
DockedTo (string)	Object ID of pod that this pod is docked to (null = not docked)
SerialNum (string)	Serial number of pod
Location (nbLocationData)	Location data for pod (see nbLocationData class)
Product (nbProductData)	Product information data for pod (see nbProductData class)
StartupTime (DateTime)	Time that pod came “online”. This is the system startup time on the “nbBaseEnclosure” pod.
EncFolderID (string)	Object ID of configuration folder for pod
UnplugAlertSev (nbErrorStatus)	Severity of alert to be generated if the pod is unplugged (nbErrorStatus.None means no alert should be generated). See nbErrorStatus enumeration.
ErrorStatus (nbErrorStatus)	Current error status for pod – this is the highest severity of the active alerts associated with the pod **or** with any of its sensors

Integrated (bool)	If true, the pod is integrated with the pod that is indicated by "DockedTo" (e.g., the sensor and camera pods on the NetBotz 455)
PortID (string)	The object ID of the port used to attach the pod to its parent (if the pod is attached to a port)
Description: The nbPod class is used to describe both the base appliance and any of the physical or logical devices attached to or monitored by the appliance. This includes physical pods (like sensor pods), logical pods (like the integrated sensors on a NetBotz 455), network-monitored devices (like SNMP crawlers), and other devices. Every appliance has at least one nbPod instance with the object ID "nbBaseEnclosure". This special nbPod represents the base appliance itself and is always defined. Other pods with specific IDs will often be present (i.e., "nbOwlEnc_0" for the docked camera on a WallBotz 500), but these IDs are not always present (and if pods are added and removed frequently, the IDs may not be in sequence), so other pod IDs should not be considered consistent between appliances.	

3.2.5 nbSensor – abstract class for sensors

Class: nbSensor	Parent Class: nbObject
Fields	
Label (string)	Label for sensor (does not include port, if any)
PodID (string)	Object ID of nbPod that contains the sensor (all sensors must be contained within an nbPod)
Location (nbLocationData)	Location information for the sensor. If null, sensor's location is same as that of its nbPod. See nbLocationData class.
ErrorStatus (nbErrorStatus)	Current error status for sensor – this is the highest severity of the active alerts associated with the sensor

PortID (string)	Object ID of the port used to connect the sensor to its pod (if any). If null, no port is used to connect the sensor.
SuggestedThreshClass (string)	Object ID of the nbThresholdType that is suggested for defining thresholds for this sensor. See nbThresholdType class.
IsMonitored (bool)	If true, one or more active thresholds are monitoring the sensor. If false, no thresholds are active.
Description: The nbSensor class is the abstract base class for all sensors. All instances of sensors will be one of the subclasses of this class (nbNumSensor, nbStateSensor, or nbStringSensor). All sensors are required to be contained within an nbPod object, so the *PodID* field is always defined and valid.	

3.2.6 nbNumSensor – class for all numeric sensors

Class: nbNumSensor	Parent Class: nbSensor
Fields	
Value (double)	Current value of the sensor. If undefined, the value is "NaN" (test with Double.IsNaN()).
UnitsID (string)	Object ID of units used for sensor value, if any. IDs are consistently defined between appliances. If null, value has no units.
AltUnitsID (string)	Alternate units (English versus Metric) available for this sensor. If null, no alternate units.
MinValue (double)	Minimum valid value for sensor. If undefined, value is "NaN". Units are same as for *Value* field.
MaxValue (double)	Maximum valid value for sensor. If undefined, value is "NaN". Units are same as for *Value* field.

ValueInc (double)	Suggested value increment (minimum change between presented values). If undefined, value is "NaN".
HistoryTime (int)	Configured value history time, in seconds. This is the maximum duration of time that the value history for the sensor will be remembered.
MaxHistoryTime (int)	Maximum allowed value for the *HistoryTime* field, in seconds
ValueFmt (string)	printf()-style format string suggested for printing the sensor value
Description: The nbNumSensor class is used to represent any numeric sensors. Since most numeric sensors have values requiring units, and unit systems are different among different countries, the values and ranges reported in the nbNumSensor object can be dependent upon the preferred unit system requested when the nbNumSensor object is read from the appliance (typically controlled by a nbUnitsRequest-typed parameter). Since the *Value* and *ErrorStatus* fields are the most likely to be monitored by repeated reading, it is suggested that the getNumSensorValues() and getErrorStatus() methods be used except where the other data is also needed (to reduce system and network traffic).	

3.2.7 nbStateSensor – class for all state sensors

Class: nbStateSensor	Parent Class: nbSensor
Fields	
ValueIndex (int)	Index of the current value of the sensor (relative to the *ValueEnum* array). If undefined, the value is -1.
ValueEnum (string[])	Array of string containing the presentation labels for each of the allowed states for the sensor (i.e., non-negative values of *ValueIndex*). Application code **must** always check that the ValueIndex value is >= 0 before indexing into this array.

HistoryTime (int)	Configured value history time, in seconds. This is the maximum duration of time that the value history for the sensor will be remembered.
MaxHistoryTime (int)	Maximum allowed value for the *HistoryTime* field, in seconds

Description: The nbStateSensor class is used to represent any state sensors. By definition, the possible values of a state sensor are required to be a finite, zero-indexed list, so the *ValueEnum* array is always defined. When the value of the sensor is unknown, the *ValueIndex* will be -1. Consequently, presentation of the value of the sensor must always test for this before indexing into the *ValueEnum* array:

```
if(x.ValueIndex < 0)
      return "N/A";
else
      return x.ValueEnum[x.ValueIndex];
```

Since the *ValueIndex* and *ErrorStatus* fields are the most likely to be monitored by repeated reading, it is suggested that the getStateSensorValues() and getErrorStatus() methods be used except where the other data is also needed (to reduce system and network traffic).

Also, nbButton and nbSwitch output controls are subclasses of nbStateSensors (for them, the value of the state sensor indicates the current state of the output), so duplicate IDs will be found when listing all state sensors and all buttons or switches for a given pod.

3.2.8 nbStringSensor – class for all string sensors

Class: nbStringSensor	Parent Class: nbSensor
Fields	
Value (string)	Current value of the string sensor. If undefined, value is null.

Description: The nbStringSensor class is used to represent any string valued sensors.

Since the *Value* and *ErrorStatus* fields are the most likely to be monitored by repeated reading, it is suggested that the getStringSensorValues() and getErrorStatus() methods be used except where the other data is also needed (to reduce system and network traffic).

3.2.9 nbPort – abstract class for all ports

Class: nbPort	Parent Class: nbObject
Fields	
State (nbPortState)	Current state of port. See nbPortState enumeration.
Label (string)	Label for port
PodID (string)	Object ID of pod containing the port. All ports are required to be contained by a pod.
Description: The nbPort class is an abstract base class for representing different I/O ports. Currently, only the nbDINPort subclass is defined.	

3.2.10 nbDINPort – class for sensor ports

Class: nbDINPort	Parent Class: nbPort
Fields	
SupportsRMS (bool)	Port supports RMS-style sensors (i.e., Amp Detect sensors) if set to “true”
SupportsAverage (bool)	Port supports average-voltage sensors (i.e., temperature, humidity, analog, 0-5V) if set to “true”
SupportsDryContact (bool)	Port supports dry-contact sensors if set to “true”
SensorIDSuffix (string)	Suggested suffix for creating a unique sensor ID for this port (generally, this is the index of the port on its pod)
Description: The nbDINPort class is an nbPort subclass used to represent DIN ports, as well as other logically similar sensor ports (dry contact ports, 4-20mA ports).	

3.2.11 nbScheduleTimes – time-of-day/day-of-week schedule

Class: nbScheduleTimes	Parent Class: none
Fields	
Sunday (nbScheduleTimes[])	Array of times on Sunday that schedule is active (15-minute intervals). See nbScheduleTimes enumeration.
Monday (nbScheduleTimes[])	Array of times on Monday that schedule is active (15-minute intervals). See nbScheduleTimes enumeration.
Tuesday (nbScheduleTimes[])	Array of times on Tuesday that schedule is active (15-minute intervals). See nbScheduleTimes enumeration.
Wednesday (nbScheduleTimes[])	Array of times on Wednesday that schedule is active (15-minute intervals). See nbScheduleTimes enumeration.
Thursday (nbScheduleTimes[])	Array of times on Thursday that schedule is active (15-minute intervals). See nbScheduleTimes enumeration.
Friday (nbScheduleTimes[])	Array of times on Friday that schedule is active (15-minute intervals). See nbScheduleTimes enumeration.
Saturday (nbScheduleTimes[])	Array of times on Saturday that schedule is active (15-minute intervals). See nbScheduleTimes enumeration.
Description: The nbSchedule class is used to represent a set of times-of-day and days-of-week that a given process or event will be active. Specifically, each of the seven days of the week is divided into 15-minute periods, and the nbScheduleTimes array for each day contains those 15-minute periods that are considered to be active.	

3.2.12 nbButton – class for button-style output controls

Class: nbButton	Parent Class: nbStateSensor
Fields	
ActivationSchedule (nbSchedule)	Schedule controlling when the button will be automatically activated. At the beginning of any of the time periods defined by the schedule, the button will be automatically "pressed". See the nbSchedule class.
AccessControlList (string[])	List of object IDs of the non-administrator privilege users that may activate the button.
Description: The nbButton class is a subclass of nbStateSensor used to represent output controls configured for a button-like behavior.	

3.2.13 nbSwitch – class for switch-style output controls

Class: nbSwitch	Parent Class: nbStateSensor
Fields	
SetToClosedSchedule (nbSchedule)	Schedule controlling when the switch will be automatically set to its "closed" state (if not already in that state). At the beginning of any of the time periods defined by the schedule, the switch will be automatically set to "closed". See the nbSchedule class.
SetToOpenSchedule (nbSchedule)	Schedule controlling when the switch will be automatically set to its "open" state (if not already in that state). At the beginning of any of the time periods defined by the schedule, the switch will be automatically set to "open". See the nbSchedule class.
AccessControlList (string[])	List of object IDs of the non-administrator privilege users that may set the state of the switch.
Description: The nbSwitch class is a subclass of nbStateSensor used to represent output controls configured for a switch-like behavior.	

3.2.14 nbAlert – class for all alerts

Class: nbAlert	Parent Class: nbObject
Fields	
Label (string)	Short description label for alert
Description (string[])	Full description text for alert (one string for each paragraph)
AlertTypeID (string)	Object ID of alert type. IDs are consistent between different appliances.
PodID (string)	Object ID of nbPod associated with alert, if any
SensorID (string)	Object ID of nbSensor associated with alert, if any
PortID (string)	Object ID of nbPort associated with alert, if any
ProfileID (string)	Object ID of nbAlertProfile associated with the alert. This is the profile that will determine which actions are executed for the alert.
ThresholdID (string)	Object ID of the nbThreshold associated with the alert, if any
Severity (nbAlertSeverity)	Severity of the alert. See the nbAlertSeverity enumeration.
StartTime (DateTime)	Time that the alert was created (i.e., the problem started)
ResolveTime (DateTime)	Time that the alert was resolved. Field value is only valid if the *IsResolved* field is “true”.
IsResolved (bool)	If “true”, the alert has been resolved. If “false”, the alert is active.

NotifyList (string[])	List of notification targets (e-mail addresses) tied to the alert. These are supplied by the "Threshold-specific e-mail addresses" in the threshold definitions.
CameraList (string[])	List of object IDs for the nbCamera objects to be triggered when processing the alert.
WasCancelled (bool)	If "true", alert was cancelled by a system restart.
Latitude (double)	Latitude of location when alert occurred, if available (negative = south, positive = north, in degrees). If not available, value is "NaN" (test using Double.IsNaN()).
Longitude (double)	Longitude of location when alert occurred, if available (negative=west, positive=east, in degrees). If not available, value is "NaN" (test using Double.IsNaN()).
URL (string)	URL associated with alert, if any. Supplied by "User URL" field in the threshold.
UserDescription (string[])	User-supplied description strings for alert. Supplied by the "User Description" fields in the threshold.
IsManualResolve (bool)	If "true", the alert requires manual resolution. This is done using the acknowledgeAlert() method.
ManualResolveUserID (string)	User ID provided when the alert was manually resolved, if any
ManualResolveNote (string)	Comment or description provided by user when alert was manually resolved, if any

Description: The nbAlert class is used to represent any alert on the appliance. Several of the fields are only valid under certain conditions:

- The *ResolveTime* field is only valid if the alert is resolved (*IsResolved*=true)
- The *ManualResolveUserID* and *ManualResolveNote* fields are only valid if the alert is manually resolved and has been resolved (*IsManualResolve*=true AND *IsResolved*=true)
- The acknowledgeAlert() API can only be used if the alert is manually resolved and is still active (*IsManualResolve*=true AND *IsResolved*=false)
- The *Label* and *Description* fields will be refreshed when the alert becomes resolved

3.2.15 nbCapturedGraph – class for captured graphs tied to alerts

Class: nbCapturedGraph	Parent Class: nbObject
Fields	
Created (DateTime)	Timestamp when graph was created
StartTime (DateTime)	Timestamp of beginning of time range presented in the graph
EndTime (DateTime)	Timestamp of end of time range presented in the graph
SensorID (string)	Object ID of sensor whose value is being graphed
XRes (int)	Horizontal resolution of graph image, in pixels
YRes (int)	Vertical resolution of graph image, in pixels
Label (string)	Label describing the captured graph
AlertID (string)	Object ID of the alert that triggered the graph capture
TotalSize (long)	Total size of captured graph image, in bytes
GraphPNGURL (string)	URL for requesting a captured graph from the appliance, as a PNG file
Description: This class is used to represent graphs captured during the processing of an alert. The object includes a URL that can be used to request a PNG-formatted representation of the captured graph from the appliance.	

3.2.16 nbCapturedPicture – single image data within a picture set

Class: nbCapturedPicture	Parent Class: none
Fields	
CapturedTime (DateTime)	Timestamp when picture was captured
CapturedTimeMsec (int)	Fractional second of capture time (milliseconds, from 0 to 999)
PictureJPEGURL (string)	URL for requesting the JPEG for the image from the appliance
Description: This class is used to represent individual picture frames within a picture set (nbPictureSet class). The URL field provides a way to load the image from the appliance.	

3.2.17 nbCapturedPictureSet – class for captured picture sets tied to alerts

Class: nbCapturedPictureSet	Parent Class: nbObject
Fields	
Created (DateTime)	Timestamp when picture set was created
StartTime (DateTime)	Timestamp of first image in picture set
EndTime (DateTime)	Timestamp of last image in picture set
CameraID (string)	Object ID of camera used to capture the picture set
XRes (int)	Horizontal resolution of images in picture set, in pixels
YRes (int)	Vertical resolution of images in picture set, in pixels
Mode (string)	ID of video mode used to capture images in picture set. This corresponds to the value of the *ID* field in the *Modes* array for the given camera.

Label (string)	Label describing the captured picture set
AlertID (string)	Object ID of the alert that triggered the picture set capture
TotalSize (long)	Total size of captured picture set, in bytes
AllPicturesJPEGURL (string)	URL for requesting a multipart/x-mixed-replace format stream of all the JPEG files in the picture set
Pictures (nbCapturedPicture[])	Array of objects representing the individual images of the picture set
Description: This class is used to represent a captured picture set captured during the processing of an alert. The object includes a URL that can be used to request a stream of the JPEG-formatted images of the picture set from the appliance. Each picture record also has a URL to load the individual JPEG from the appliance.	

3.2.18 nbNumSensorHistory – container for value update history for a numeric sensor

Class: nbNumSensorHistory	Parent Class: none
Fields	
Now (DateTime)	Timestamp for current time when history object was created. All other times are relative to this.
Values (double[])	List of value updates, from most recent to least recent, in the time range requested. For intervals when the value was unavailable, the value will be "NaN" (test using Double.IsNaN()).
ValueChangedSecAgo (double[])	List of relative timestamps indicating the number of seconds in the past (relative to *Now* field) that the value of the sensor changed to the value in the *Values* array with the same index. Note: The last element in the array will often correspond to a time before the time interval requested, as that value update is what determines the value of the sensor as of the beginning of the requested range. Note: When used in the getNumSensorValueHistoryWithFixedInterval( ) method, the value returned is simply the time corresponding to the time of the next fixed interval in the requested range.
Description: This class is used to contain a set of value updates returned by the getNumSensorValueHistory() and getNumSensorValueHistoryWithFixedInterval() methods.	

3.2.19 nbNumSensorHistoryUTC – container for value update history for a numeric sensor using UTC timestamps

Class: nbNumSensorHistoryUTC	Parent Class: none
Fields	
Now (DateTime)	Timestamp for current time when history object was created. All other times are relative to this.
Values (double[])	List of value updates, from most recent to least recent, in the time range requested. For intervals when the value was unavailable, the value will be "NaN" (test using Double.IsNaN()).
ValueTimes (DateTime[])	List of absolute timestamps indicating the times that the value of the sensor changed to the value in the *Values* array with the same index. Note: The last element in the array will often correspond to a time before the time interval requested, as that value update is what determines the value of the sensor as of the beginning of the requested range. For fixed interval requests (Interval>0.0), the times simply correspond to the next fixed interval within the range.
Description: This class is used to contain a set of value updates returned by the getNumSensorValueHistoryUTC() method.	

3.2.20 nbStateSensorHistory – container for value update history for a state sensor

Class: nbStateSensorHistory	Parent Class: none
Fields	
Now (DateTime)	Timestamp for current time when history object was created. All other times are relative to this.
Values (int[])	List of value updates, from most recent to least recent, in the time range requested. Values correspond to the *ValueIndex* field in the nbStateSensor class. For intervals when the value was unavailable, the value will be -1.
ValueChangedSecAgo (double[])	List of relative timestamps indicating the number of seconds in the past (relative to *Now* field) that the value of the sensor changed to the value in the *Values* array with the same index. Note: The last element in the array will often correspond to a time before the time interval requested, as that value update is what determines the value of the sensor as of the beginning of the requested range. Note: When used in the getStateSensorValueHistoryWithFixedInterval( ) method, the value returned is simply the time corresponding to the time of the next fixed interval in the requested range.
Description: This class is used to contain a set of value updates returned by the getStateSensorValueHistory() and getStateSensorValueHistoryWithFixedInterval() methods.	

3.2.21 nbStateSensorHistoryUTC – container for value update history for a state sensor with UTC timestamps

Class: nbStateSensorHistoryUTC	Parent Class: none
Fields	
Now (DateTime)	Timestamp for current time when history object was created. All other times are relative to this.
Values (int[])	List of value updates, from most recent to least recent, in the time range requested. Values correspond to the *ValueIndex* field in the nbStateSensor class. For intervals when the value was unavailable, the value will be -1.
ValueTimes (DateTime[])	List of absolute timestamps indicating the times that the value of the sensor changed to the value in the *Values* array with the same index. Note: The last element in the array will often correspond to a time before the time interval requested, as that value update is what determines the value of the sensor as of the beginning of the requested range. For fixed interval requests (Interval>0.0), the times simply correspond to the next fixed interval within the range.
Description: This class is used to contain a set of value updates returned by the getStateSensorValueHistoryUTC() method.	

3.2.22 nbCameraMode – class describing a specific camera mode

Class: nbCameraMode	Parent Class: none
Fields	
ID (string)	Mode identification string
Label (string)	Descriptive label for mode
XRes (int)	Horizontal resolution of images generated in this mode, in pixels
YRes (int)	Vertical resolution of images generated in this mode, in pixels
PanXRange (int)	Horizontal range for digital panning, in pixels
PanYRange (int)	Vertical range for digital tilting, in pixels
CaptureImageURL (string)	URL to use for requesting a single JPEG image captured in this mode
CaptureImageStreamURL (string)	URL to use for requesting an image stream (JPEGs returned as a multipart/x-mixed-replace stream ("server-push")). Append "interval=*N*" parameter (where *N* is interval between images in milliseconds) to request specific frame rate.
Description: This class is used to describe a specific video mode supported by a camera.	

3.2.23 nbCameraQualityLevel – class describing a specific camera compression/quality level

Class: nbCameraQualityLevel	Parent Class: none
Fields	
ID (string)	Quality level identification string
Label (string)	Descriptive label for quality level
MaxFrameRatePerMode (double[])	Array of maximum frame rates supported at this quality level, indexed by the video mode being captured (index corresponding to the nbCameraMode array for the camera)
TypicalImageSize (int[])	Array of typical image sizes for pictures generated with this quality level, indexed by the video mode being captured (index corresponding to the nbCameraMode array for the camera)
Description: This class is used to describe a specific image compression/quality mode supported by a camera.	

3.2.24 nbCameraColorBalancePreset – class describing a specific camera color balance preset

Class: nbCameraColorBalancePreset	Parent Class: none
Fields	
Label (string)	Descriptive label for preset
BlueColorBalance (int)	Blue color balance for preset (1-255)
RedColorBalance (int)	Red color balance for preset (1-255)
Description: This class is used to describe a specific standard color balance preset supported by a camera.	

3.2.25 nbCamera – class representing a camera and its capabilities and settings

Class: nbCamera	Parent Class: nbObject
Fields	
PodID (string)	Object ID of pod containing the camera
Label (string)	Label for the camera
FullLabel (string)	Full label for camera (pod + camera). This label is suggested for most uses.
Modes (nbCameraMode[])	Array of video modes supported by the camera. See nbCameraMode class for details.
DefaultModeIndex (int)	Index in the *Modes* array for the default video mode
DefCaptureImageURL (string)	URL for capturing an image from the camera using its default video mode, in JPEG format
DefCaptureImageStreamURL (string)	URL to use for requesting an image stream (JPEGs returned as a multipart/x-mixed-replace stream ("server-push")) of images in the default mode. Append "interval=*N*" parameter (where *N* is interval between images in milliseconds) to request specific frame rate.
QualityLevels (nbCameraQualityLevel[])	Array of available image quality/compression levels. See nbCameraQualityLevel class for details.
QualityIndex (int)	Index in *QualityLevels* array for currently configured quality level
Brightness (int)	Current brightness setting (0-255, 100 is default)
InvertImage (bool)	Setting for inverting the camera image (true=inverted, false=normal)

ColorBalance (nbCameraColorBalance)	Current color balance option. See nbCameraColorBalance enum for details.
CustomBlueColorBalance (int)	If *ColorBalance*=CustomColorBalance, this is the blue color balance setting that has been configured.
CustomRedColorBalance (int)	If *ColorBalance*=CustomColorBalance, this is the red color balance setting that has been configured
GammaCorrection (double)	Current gamma correction setting for the camera
IsFullView (bool)	Determines if camera will show full field of view (true) or "pan-and-scan" (false)
AudioInID (string)	Object ID of audio input associated with the camera, if any
AudioOutID (string)	Object ID of audio output associated with the camera, if any
MaxInteractiveRatePct (int)	Percentage of maximum frame rate that interactive users should be allowed to request (1-100)
CaptureModeIndex (int)	Index (relative to *Modes* array) of video mode configured to be used for capturing picture sets
CapturePanX (int)	Horizontal pan position for captured images, in pixels
CapturePanY (int)	Vertical pan position for captured images, in pixels
CaptureNumPix (int)	Total number of pictures to capture in each picture set
CaptureIntervalMsec (int)	Target interval between captured pictures, in milliseconds
CaptureDelayMsec (int)	Delay before capture of first picture in a

	picture set, in milliseconds
CapturePreEventPix (int)	Number of pictures to capture before capture triggered. Must be less than *CaptureNumPix*.
CameraMotionEnabled (bool)	If “true”, camera motion sensor is enabled
CameraMotionSensitivity (int)	Camera motion sensor sensitivity setting
CameraMotionArea (int)	Camera motion sensor area of motion setting
CameraMotionOutline (bool)	Camera motion outlining option (true=enabled)
CameraMotionMask (byte[])	Bitmask for controlling which patches on the camera are monitored by the camera motion sensor. Bits are ordered low-to-high per byte, left to right, top-to-bottom. The number of bits is (CameraMotionXRes / CameraMotionXPatchRes) * (CameraMotionYRes / CameraMotionYPatchRes).
CameraMotionXRes (int)	Horizontal resolution of area observed by camera motion sensor, in pixels.
CameraMotionYRes (int)	Vertical resolution of area observed by camera motion sensor, in pixels
CameraMotionXPatchRes (int)	Horizontal size of each patch monitored by camera motion sensor, in pixels
CameraMotionYPatchRes (int)	Vertical size of each patch monitored by camera motion sensor, in pixels
CameraMotionOutlineSupported (bool)	If true, camera motion outlining is supported (*CameraMotionOutline* field)
CameraMotionMaskSupported (bool)	If true, camera motion mask is supported (*CameraMotionMask* field)

CameraMotionSensorID (string)	Object ID of camera motion sensor associated with the camera, if any
Description: This class is used to represent each of the cameras (or other video input devices) associated with the appliance. Each nbCamera object describes the capabilities and configuration of a single video input source. Any nbCamera will be contained within a pod.	

3.2.26 nbNumericThresholdParm – class defining data specific to numeric parameters

Class: nbNumericThresholdParm	Parent Class: none
Fields	
Value (double)	Value of parameter setting. This field can be updated when creating or updating a threshold.
MinValue (double)	Minimum value allowed for parameter setting (*Value* field)
MaxValue (double)	Maximum value allowed for parameter setting (*Value* field)
ValueInc (double)	Suggested increment between consecutive values (*Value* field)
UnitsID (string)	Object ID of units for parameter, if any
UnitsLabel (string)	Units label for parameter, if any
MaxFromParmID (string)	If defined, *Value* field must be set to a value less than or equal to the *Value* field of the parameter with this *ID*
MinFromParmID (string)	If defined, *Value* field must be set to a value greater than or equal to the *Value* field of the parameter with this *ID*
Description: This class defines the fields for a parameter that are specific to those parameters that are numeric (FloatingPointValue). When updating or creating a threshold, only the *Value* field can be updated by the application.	

3.2.27 nbIntegerThresholdParm – class defining data specific to integer parameters

Class: nbIntegerThresholdParm	Parent Class: none
Fields	
Value (int)	Value of parameter (if *ChoiceLabels*=null), or index of value (otherwise) for parameter. This field can be updated when creating or updating a threshold.
ChoiceLabels (string[])	List of labels for the valid values of the parameter. If null, parameter is a pure integer value instead of an index.
MinValue (int)	Minimum value allowed for parameter setting (*Value* field)
MaxValue (int)	Maximum value allowed for parameter setting (*Value* field)
UnitsID (string)	Object ID of units for parameter, if any
UnitsLabel (string)	Units label for parameter, if any
Description: This class defines the fields for a parameter that are specific to those parameters that are integer-typed (IntegerValue). This includes both enumeration type parameters (where ChoiceLabels is defined) and simple integer values (where ChoiceLabels is null). When updating or creating a threshold, only the *Value* field can be updated by the application.	

3.2.28 nbStringThresholdParm – class defining data specific to string parameters

Class: nbStringThresholdParm	Parent Class: none
Fields	
Value (string)	Value of the parameter. If *Choices* field is defined, this must be the value of one of the strings from this array. When creating or updating a threshold, this field may be updated.
Choices (string[])	If defined, this is the list of values that may be used for the *Value* field. They should **not** be used for presentation unless the *ChoiceLabels* field is undefined.
ChoiceLabels (string[])	If defined, these are the presentational labels associated with each of the values in the *Choices* array. The values in this array should **never** be used in the *Value* field.
MaxLength (int)	Maximum number of characters to allow in *Value* field

Description: This class defines the fields for a parameter that are specific to those parameters that are string-typed (StringValue). This includes both enumeration type parameters (where *Choices* is defined) and simple string values (where *Choices* is null).

If *Choices* is defined, *Value* must be set to **one** of the strings in the presented list, or null. If *ChoiceLabels* is defined, the strings in the *ChoiceLabels* are what should be presented to the user (as opposed to the *Choices* strings). Even when *ChoiceLabels* is defined, the strings from *Choices* must be used for the *Value* field.

When updating or creating a threshold, only the *Value* field can be updated by the application.

3.2.29 nbStringListThresholdParm – class defining data specific to string list parameters

Class: nbStringListThresholdParm	Parent Class: none
Fields	
Value (string[])	Value of the parameter. If *Choices* field is defined, this must be an array of zero or more of the strings from this array. When creating or updating a threshold, this field may be updated.
Choices (string[])	If defined, this is the list of values that may be used for the *Value* field. They should **not** be used for presentation unless the *ChoiceLabels* field is undefined.
ChoiceLabels (string[])	If defined, these are the presentational labels associated with each of the values in the *Choices* array. The values in this array should **never** be used in the *Value* field.
MaxLength (int)	Maximum number of characters to allow in *Value* field

Description: This class defines the fields for a parameter that are specific to those parameters that are string-list-typed (StringListValue). This includes both multi-select enumeration type parameters (where *Choices* is defined) and simple multi-line string values (where *Choices* is null).

If *Choices* is defined, *Value* must be set to an array of zero or more of the strings in the presented list. If *ChoiceLabels* is defined, the strings in the *ChoiceLabels* are what should be presented to the user (as opposed to the *Choices* strings). Even when *ChoiceLabels* is defined, the strings from *Choices* must be used for the *Value* field.

If Choices is null, the *Value* field should be considered a multi-line field, with one string for each line.

When updating or creating a threshold, only the *Value* field can be updated by the application.

3.2.30 nbBooleanThresholdParm – class defining data specific to boolean parameters

Class: nbBooleanThresholdParm	Parent Class: none
Fields	
Value (bool)	Value of parameter. If threshold is being updated or created, this field can be updated.
ChoiceLabels (string[])	List of labels for the valid values of the parameter. If null, parameter is a pure boolean value (logical check box). If set, the ChoiceLabels[0] is false and ChoiceLabels[1] is true.
Description: This class defines the fields for a parameter that are specific to those parameters that are boolean-typed (BooleanValue). This includes both enumeration type parameters (where ChoiceLabels is defined) and simple checkbox-type values (where ChoiceLabels is null). When updating or creating a threshold, only the *Value* field can be updated by the application.	

3.2.31 nbScheduleThresholdParm – class defining data specific to schedule parameters

Class: nbScheduleThresholdParm	Parent Class: none
Fields	
Value (nbSchedule)	Value of parameter. If threshold is being updated or created, this field can be updated. See nbSchedule class for details.
Description: This class defines the fields for a parameter that are specific to those parameters that are schedule-typed (ScheduleValue). When updating or creating a threshold, only the *Value* field can be updated by the application.	

3.2.32 nbThresholdType – class describing each type of threshold available

Class: nbThresholdType	Parent Class: nbObject
Fields	
Label (string)	Presentation label for the type of threshold
SensorType (nbThresholdSensorType)	Type of sensor supported by the threshold type. See nbThresholdSensorType enumeration for details.
Description: This class is used to provide details of the different types of thresholds supported by the appliance, as well as which sensor types are supported by the threshold type.	

3.2.33 nbThresholdParm – class describing a single parameter of a threshold

Class: nbThresholdParm	Parent Class: none
Fields	
Label (string)	Presentation label for the parameter
IsAdvanced (bool)	Indicates whether the parameter is considered advanced (true) or basic (false). Used for user interface.
Type (nbThresholdParmType)	Indicates the general data type for the threshold. This also determines which of the **Parm* fields will be defined.
SubType (string)	Indicates the subtype of the field, which can allow further user interface optimization. "Well known" subtypes include: • "notifylist" – values are e-mail addresses • "http_url_macros" – value is a URL, and supports macros • "multiline_macros" – value is multi-line text, and supports macros • "rate" – value is a rate-of-change, so ranges from sensor do not apply
FloatParm (nbNumericThresholdParm)	If *Type*=FloatingPointValue, this field will be defined to contain the data particular to a floating point typed parameter. See nbNumericThresholdParm class for details.
IntParm (nbIntegerThresholdParm)	If *Type*=IntegerValue, this field will be

	defined to contain the data particular to an integer typed parameter. See nbIntegerThresholdParm class for details.
StringParm (nbStringThresholdParm)	If *Type*=StringValue, this field will be defined to contain the data particular to a string typed parameter. See nbStringThresholdParm class for details.
BoolParm (nbBooleanThresholdParm)	If *Type*=BooleanValue, this field will be defined to contain the data particular to a boolean typed parameter. See nbBooleanThresholdParm class for details.
SchedParm (nbScheduleThresholdParm)	If *Type*=ScheduleValue, this field will be defined to contain the data particular to a schedule typed parameter. See nbScheduleThresholdParm class for details.
StringListParm (nbStringListThresholdParm)	If *Type*=StringListValue, this field will be defined to contain the data particular to a string list typed parameter. See nbStringListThresholdParm class for details.
Description: This class is used to represent a single parameter definition and value for an nbThreshold. When creating or updating a threshold, all the data in this class (and most of the data in the appropriate **Parm* elements) must not be modified. The fields that need to be updated are defined in the appropriate class definitions for the different **Parm* fields. For any parameter, exactly one of the **Parm* fields will be non-null, and this will correspond to the field appropriate to the *Type* field value.	

3.2.34 nbThreshold – class describing a threshold or threshold template

Class: nbThreshold	Parent Class: nbObject
Fields	
Label (string)	Presentation label for the threshold. Applications should set this field when

	creating a threshold, and may update this field when updating a threshold.
ThresholdTypeID (string)	Object ID of the threshold type used to define the class. See nbThresholdType class.
SensorID (string)	Object ID of the sensor associated with the threshold
UnitsReq (nbUnitsRequest)	This indicates the units system that was selected as preferred when the nbThreshold object was generated.
Parms (nbThresholdParm[])	Array of the parameters defined for the threshold. The array should be considered ordered, when presenting the information on a GUI. Also, applications should assume that the *ID* field of the parameters will remain consistent for a given threshold type, as opposed to the index of the parameter (i.e., software updates may add parameters earlier in the array than existing parameters).

Description: This class is used to represent a threshold or a threshold template (as returned by the getThresholdTemplate() method). If the nbThreshold is a threshold template, the fields inherited from the nbObject base class will be invalid and must be ignored. Only the *Label* field and the appropriate fields within the elements of the *Parms* array may be updated when creating or updating a threshold (see the nbThresholdParm class for details).

Applications should avoid forming dependencies on the position of the elements in the *Parms* array, as these are subject to being changed (due to adding new parameters before the existing ones). Instead, any threshold, type-specific dependencies should only be formed based on the *ID* field of the parameter (i.e., it is okay to depend on the "Enabled" parameter being the element with *ID*="enabled", but it is not okay to assume it will stay the Nrd element in the array).

3.2.35 nbAlertProfileSchedule – class describing a single schedule within an alert profile

Class: nbAlertProfileSchedule	Parent Class: none
Fields	
Label (string)	Presentation label for the schedule. Applications should always set this field.
RepeatIntervalSec (int)	Number of seconds between repeated runs of this schedule. Must be 1 or greater.
MaxRepeats (int)	Maximum number of times to repeat this schedule. Must be 0 or greater.
FirstRunDelaySec (int)	Number of seconds between when the alert being processed occurs and when the first run of the schedule should happen. Must be 0 or greater.
AlertActionIDs (string[])	Array of object IDs for the alert actions (nbAlertAction) to be run each time this schedule is executed.
CapturePictures (nbAlertProfileScheduleCaptureOpt)	Controls whether to trigger picture captures when the schedule is run for alerts that indicate cameras to trigger. See nbAlertProfileScheduleCaptureOpt enum for details. Should default to CaptureWhenNeeded.
CaptureGraphs (nbAlertProfileScheduleCaptureOpt)	Controls whether to trigger graph captures when the schedule is run for alerts that are associated with a sensor. See nbAlertProfileScheduleCaptureOpt enum for details. Should default to CaptureWhenNeeded.
EnableSchedule (nbSchedule)	Controls what time-of-day/day-of-week the schedule is considered to be enabled. When not enabled, any scheduled runs

	are not done. See nbSchedule class for details. Default should be all enabled.
IsDefaultSchedule (bool)	This flag, if set to “true”, indicates that any new alert action, when created, should automatically be added to the *AlertActionIDs* list for this schedule.

Description: This class describes the details of a single alert profile schedule within an alert profile. Specifically, it defines a set of alert actions to be run, how long after an alert is created they should be run, and how often and how many times to repeat the run. If an alert is resolved, no further execution of the schedule will occur.

The CapturePictures and CaptureGraphs fields allow the schedule to force pictures or graphs to be (or not to be) collected during the execution of the schedule. By default, these resources are only captured when one or more of the alert actions in the schedule are executed and request them.

The EnableSchedule allows the schedule to be active during specific times of the day or days of the week.

3.2.36 nbAlertProfile – class describing an alert profile

Class: nbAlertProfile	Parent Class: nbObject
Fields	
Label (string)	Presentation label for the profile. Applications should always set this field.
Schedules (nbAlertProfileSchedule[])	Array of schedules describing the different alert action execution strategies for the profile. See the nbAlertProfileSchedule class for details.
DisableUntilTime (DateTime)	When set, this indicates that the alert profile should be disabled until the given time has passed.

Description: This class defines an alert profile. The appliances contain two standard profiles (both of which cannot be deleted), the default profile ("nbAlertProfile_default") and the global profile ("nbGlobalAlertProfileID"). The default profile, as its name implies, is the default profile used for processing alerts (unless the alerts specify another profile). The global profile is a special profile used for containing actions that must be run for all alerts (in addition to any actions that are run due to the profile they use).

When an alert becomes active, it indicates a specific profile to be used to process it. The schedules within the profile are each processed for the alert, and actions within those schedules are executed as needed. Once the alert is resolved, all of the actions that were run at least once during the life of the alert all run to do "return-to-normal" processing.

3.2.37 nbNumericAlertActionParm – class defining data specific to numeric parameters

Class: nbNumericAlertActionParm	Parent Class: none
Fields	
Value (double)	Value of parameter setting. This field can be updated when creating or updating an alert action.
MinValue (double)	Minimum value allowed for parameter setting (*Value* field)
MaxValue (double)	Maximum value allowed for parameter setting (*Value* field)
ValueInc (double)	Suggested increment between consecutive values (*Value* field)
UnitsID (string)	Object ID of units for parameter, if any
UnitsLabel (string)	Units label for parameter, if any
MaxFromParmID (string)	If defined, *Value* field must be set to a value less than or equal to the *Value* field of the parameter with this *ID*
MinFromParmID (string)	If defined, *Value* field must be set to a value greater than or equal to the *Value* field of the parameter with this *ID*
Description: This class defines the fields for a parameter that are specific to numeric (FloatingPointValue) parameters. When updating or creating an alert action, only the *Value* field can be updated by the application.	

3.2.38 nbIntegerAlertActionParm – class defining data specific to integer parameters

Class: nbIntegerAlertActionParm	Parent Class: none
Fields	
Value (int)	Value of parameter (if *ChoiceLabels*=null) or index of value for parameter. This field can be updated when creating or updating an alert action.
ChoiceLabels (string[])	List of labels for the valid values of the parameter. If null, parameter is a pure integer value instead of an index.
MinValue (int)	Minimum value allowed for parameter setting (*Value* field)
MaxValue (int)	Maximum value allowed for parameter setting (*Value* field)
UnitsID (string)	Object ID of units for parameter, if any
UnitsLabel (string)	Units label for parameter, if any
Description: This class defines the fields for a parameter that are specific to those parameters that are integer-typed (IntegerValue). This includes both enumeration type parameters (where ChoiceLabels is defined) and simple integer values (where ChoiceLabels is null). When updating or creating an alert action, only the *Value* field can be updated by the application.	

3.2.39 nbStringAlertActionParm – class defining data specific to string parameters

Class: nbStringAlertActionParm	Parent Class: none
Fields	
Value (string)	Value of the parameter. If *Choices* field is defined, this must be the value of one of the strings from this array. When creating or updating an alert action, this field may be updated.
Choices (string[])	If defined, this is the list of values that may be used for the *Value* field. They should **not** be used for presentation unless the *ChoiceLabels* field is undefined.
ChoiceLabels (string[])	If defined, these are the presentational labels associated with each of the values in the *Choices* array. The values in this array should **never** be used in the *Value* field.
MaxLength (int)	Maximum number of characters to allow in *Value* field.
IsPassword (bool)	If set to "true," the value of this parameter is a password. This means that the value should be hidden while being entered (and confirmation should be done), and that the *Value* found when the alert action is received from the appliance is a special value that is NOT the actual value.
Description: This class defines the fields for a parameter that are specific to those parameters that are string-typed (StringValue). This includes both enumeration type parameters (where *Choices* is defined) and simple string values (where *Choices* is null). If *Choices* is defined, *Value* must be set to ONE of the strings in the presented list, or	

null. If *ChoiceLabels* is defined, the strings in the *ChoiceLabels* are what should be presented to the user (as opposed to the *Choices* strings). Even when *ChoiceLabels* is defined, the strings from *Choices* must be used for the *Value* field.

When updating or creating an alert action, only the *Value* field can be updated by the application.

If the *IsPassword* field is set, several special factors must be considered:

- The string in the *Value* field will not contain the actual value, when read from the appliance. The field is "write-only". The application should only modify the *Value* field when attempting to change the password.

- Input of the field within the application should be hidden and should have some form of verification.

3.2.40 nbStringListAlertActionParm – class defining data specific to string list parameters

Class: nbStringListAlertActionParm	Parent Class: none
Fields	
Value (string[])	Value of the parameter. If the *Choices* field is defined, this must be an array of zero or more of the strings from this array. When creating or updating an alert action, this field may be updated.
Choices (string[])	If defined, this is the list of values that may be used for the *Value* field. They should **not** be used for presentation unless the *ChoiceLabels* field is undefined.
ChoiceLabels (string[])	If defined, these are the presentational labels associated with each of the values in the *Choices* array. The values in this array should NEVER be used in the *Value* field.
MaxLength (int)	Maximum number of characters to allow in *Value* field

Description: This class defines the fields for a parameter that are specific to those parameters that are string-list-typed (StringListValue). This includes both multi-select enumeration type parameters (where *Choices* is defined) and simple multi-line string values (where *Choices* is null).

If *Choices* is defined, *Value* must be set to an array of zero or more of the strings in the presented list. If *ChoiceLabels* is defined, the strings in the *ChoiceLabels* are what should be presented to the user (as opposed to the *Choices* strings). Even when *ChoiceLabels* is defined, the strings from *Choices* must be used for the *Value* field.

If Choices is null, the *Value* field should be considered a multi-line field, with one string for each line.

When updating or creating an alert action, only the *Value* field can be updated by the application.

3.2.41 nbBooleanAlertActionParm – class defining data specific to boolean parameters

Class: nbBooleanAlertActionParm	Parent Class: none
Fields	
Value (bool)	Value of parameter. If an alert action is being updated or created, this field can be updated.
ChoiceLabels (string[])	List of labels for the valid values of the parameter. If null, parameter is a pure boolean value (logical check box). If set, the ChoiceLabels[0] is for false and ChoiceLabels[1] is for true.
Description: This class defines the fields for a parameter that are specific to those parameters that are boolean-typed (BooleanValue). This includes both enumeration type parameters (where ChoiceLabels is defined) and simple checkbox-type values (where ChoiceLabels is null). When updating or creating an alert action, only the *Value* field can be updated by the application.	

3.2.42 nbScheduleAlertActionParm – class defining data specific to schedule parameters

Class: nbScheduleAlertActionParm	Parent Class: none
Fields	
Value (nbSchedule)	Value of parameter. If an alert action is being updated or created, this field can be updated. See nbSchedule class for details.
Description: This class defines the fields for a parameter that are specific to those parameters that are schedule-typed (ScheduleValue). When updating or creating an alert action, only the *Value* field can be updated by the application.	

3.2.43 nbAlertActionParm – class describes a single parameter of an alert action

Class: nbAlertActionParm	Parent Class: none
Fields	
Label (string)	Presentation label for the parameter.
IsAdvanced (bool)	Indicates whether the parameter is considered advanced (true) or basic (false). Used for user interface.
Type (nbAlertActionParmType)	Indicates the general data type for the parameter. This also determines which of the **Parm* fields will be defined.
SubType (string)	Indicates the subtype of the field, which can allow further user interface optimization. "Well known" subtypes include: • "notifylist" – values are e-mail addresses • "http_url_macros" – value is a URL, and supports macros. • "multiline_macros" – value is multi-line text, and supports macros.
FloatParm (nbNumericAlertActionParm)	If *Type*=FloatingPointValue, this field will be defined to contain the data particular to a floating point typed parameter. See nbNumericAlertActionParm class for details.
IntParm (nbIntegerAlertActionParm)	If *Type*=IntegerValue, this field will be defined to contain the data particular to an integer typed parameter. See nbIntegerAlertActionParm class for

	details.
StringParm (nbStringAlertActionParm)	If *Type*=StringValue, this field will be defined to contain the data particular to a string typed parameter. See nbStringAlertActionParm class for details.
BoolParm (nbBooleanAlertActionParm)	If *Type*=BooleanValue, this field will be defined to contain the data particular to a boolean typed parameter. See nbBooleanAlertActionParm class for details.
SchedParm (nbScheduleAlertActionParm)	If *Type*=ScheduleValue, this field will be defined to contain the data particular to a schedule typed parameter. See nbScheduleAlertActionParm class for details.
StringListParm (nbStringListAlertActionParm)	If *Type*=StringListValue or OrderedStringListValue, this field will be defined to contain the data particular to a string list typed parameter. See nbStringListAlertActionParm class for details.
Description: This class is used to represent a single parameter definition and value for an nbAlertAction. When creating or updating an alert action, all the data in this class (and most of the data in the appropriate **Parm* elements) must not be modified. The fields that need to be updated are defined in the appropriate class definitions for the different **Parm* fields. For any parameter, exactly one of the **Parm* fields will be non-null, and this will correspond to the field appropriate to the *Type* field value.	

3.2.44 nbAlertAction – class describes an alert action or alert action template

Class: nbAlertAction	Parent Class: nbObject
Fields	
Label (string)	Presentation label for the alert action. Applications should set this field when creating an alert action and may update this field when updating an alert action.
ActionClassID (string)	Object ID of the alert action class used to define the alert action. See the nbAlertActionClass class for details.
UnitsReq (nbUnitsRequest)	This indicates the units system that was selected as preferred when the nbAlertAction object was generated.
Parms (nbAlertActionParm[])	Array of the parameters defined for the alert action. The array should be considered ordered, when presenting the information on a GUI. Also, applications should assume that the *ID* field of the parameters will remain consistent for a given alert action class, as opposed to the index of the parameter (i.e., software updates may add parameters earlier in the array than existing parameters).
EnabledSched (nbSchedule)	Controls what times-of-day and days-of-week that this action is capable of being run. During time intervals when it is not enabled, the action will not be run by any alert profile schedules attempting to run them. Applications can modify this field and, by default, should enabled the field for all times and days. See nbSchedule class for details
ActiveForSevs (nbAlertSeverity[])	List of alert severities that the alert action should process. An alert profile schedule

	processing an alert with a severity not in this list will not execute the action. See nbAlertSeverity enum for details. Applications can set this field. The field should default to a list of all severity values (enabled for all).
CanCapturePictures (bool)	If true, the alert action is capable of processing captured pictures
CapturePicturesMaxCount (int)	If non-zero, the alert action is requesting the capture of pictures and will process up to the given number of images. Applications can set this field.
CanCaptureSound (bool)	If true, the alert action is capable of processing captured audio
CaptureSound (bool)	If true, the alert action is requesting that audio be captured along with any captured pictures. Applications can set this field.
CanCaptureGraphs (bool)	If true, the alert action is capable of processing captured graphs
CaptureGraphs (bool)	If true, the alert action is requesting that graphs be captured. Applications can set this field.
AddToDefSchedules (bool)	If set to "true", the alert action will be added to any alert profile schedules with the "IsDefaultSchedule" field set to "true". Applications can set this field.

Description: This class is used to represent an alert action or an alert action template (as returned by the getAlertActionTemplate() method). If the nbAlertAction is an alert action template, the fields inherited from the nbObject base class will be invalid, and must be ignored. Only the fields indicated as "application settable" and the appropriate fields within the elements of the *Parms* array may be updated when creating or updating an alert action (see the nbAlertActionParm class for details).

Applications should avoid forming dependencies on the position of the elements in the *Parms* array, as these are subject to being changed (due to adding new parameters

before the existing ones). Instead, any alert action class-specific dependencies should only be formed based on the *ID* field of the parameter (i.e., it is okay to depend on the "Enabled" parameter being the element with *ID*="enabled", but it is not okay to assume it will stay the Nrd element in the array).

3.2.45 nbAlertActionClass – class defining an alert action class

Class: nbAlertActionClass	Parent Class: nbObject
Fields	
Label (string)	Presentation label for the action class
Description: This class is used to represent a type of alert action (an alert action class) that can be used for creating alert actions (nbAlertAction). The object IDs for alert action classes for a given type of action are consistent between appliances that implement the action.	

3.2.46 nbUserAccount – class defining a user account

Class: nbUserAccount	Parent Class: nbObject
Fields	
Label (string)	Presentation label for user's name
UserID (string)	User ID
Password (string)	Password (write only)
PrivSetID (string)	Object ID of the privilege set controlling the privileges of the user account. If null, account has no privileges.

Description: This class is used to represent user accounts on the appliance. All fields in the class can be updated by an application, with the following restrictions:

- The *UserID* field of the "nbUser_guest" account (the guest account) is "*" and cannot be changed.
- The *Password* field is write-only, so the value read from it is meaningless, but should not be modified unless that password is intended to be changed.
- The *PrivSetID* field for the "nbUser_admin" account (the default administrator) is "nbPrivSet_admin" and cannot be changed.
- The *UserID* field for a new or updated account must be distinct from the other accounts.

3.2.47 nbUserPrivSet – class defining a user privilege set

Class: nbUserPrivSet	Parent Class: nbObject
Fields	
Label (string)	Presentation label for privilege set

Description: This class is used to represent a set of access privileges that can be assigned to user accounts. There are several "well known" privilege sets that all appliances support:

- "nbPrivSet_admin" – administrator privileges
- "nbPrivSet_application_alert" – application privileges, with alert update
- "nbPrivSet_application" – application privileges
- "nbPrivSet_sensor" – sensor privileges
- "nbPrivSet_sensornocamera" – sensor privileges, with no camera access

3.2.48 nbEmailServer – class defining a single e-mail server's settings

Class: nbEmailServer	Parent Class: none
Fields	
HostName (string)	Hostname/IP address of the server
Port (int)	Port number (25 is default)
UserID (string)	User ID for authenticating with SMTP server (blank if not needed)
Password (string)	Password for authenticating with SMTP server (write only field – when read, this will be "$$hidden$$". When written, any other value will cause the password to be updated).
UseSSL (nbUseSSLChoices)	Enum of choices as to whether or not to use SSL, and how strict to be with certificate verification.
Description: This class is used to represent the settings for one of the e-mail servers configured on the appliance. The nbEmailServerSettings.Servers field contains an ordered array of these objects, indicating the primary and backup e-mail servers.	

3.2.49 nbEmailServerSettings – e-mail server settings for the appliance

Class: nbEmailServerSettings	Parent Class: None
Fields	
FromAddress (string)	"From" address used for e-mails from the appliance. If blank, netbotz@netbotz.com is used.
Servers (nbEmailServer[])	Ordered array of e-mail server definitions. These represent the e-mail servers to be used for outbound mail, in order of preference (primary, backup, etc).
Description: This class is used to represent the outbound e-mail settings for the appliance.	

3.2.50 nbMap – class defining a user-defined map

Class: nbMap	Parent Class: nbObject
Fields	
Label (string)	Presentation label for map
XRes (int)	Horizontal resolution of map (pixels)
YRes (int)	Vertical resolution of map (pixels)
JpegURL (string)	URL that the caller may use to request a JPEG encoded image of the map
GifURL (string)	URL that the caller may use to request a GIF encoded image of the map
PngURL (string)	URL that the caller may use to request a PNG encoded image of the map
LayerURLs (string[])	Array of URLs that may be used to request each of the layers of the map, from bottom to top. Layers above the first tend to be transparent images. The last layer contains the objects (sensors, pods) that are on the map.
Description: This class is used to represent a user-defined map, and to provide the attributes and URLs needed to access the map.	

3.2.51 nbCapturedMap – class for capture maps tied to alerts

Class: nbCapturedMap	Parent Class: nbObject
Fields	
Created (DateTime)	Timestamp when graph was created
AlertingObjectID (string)	Object ID of the alerting object which triggered the capture
MapID (string)	Object ID of nbMap which was used to generate the captured map
XRes (int)	Horizontal resolution of map image, in pixels
YRes (int)	Vertical resolution of map image, in pixels
Label (string)	Label describing the captured map
AlertID (string)	Object ID of the alert that triggered the map capture
TotalSize (long)	Total size of captured map image, in bytes
MapJPEGURL (string)	URL for requesting captured map from appliance, as a JPEG file
Description: This class is used to represent maps captured during the processing of an alert. The object includes a URL that can be used to request a JPEG-formatted representation of the captured map from the appliance.	

3.3 Enumeration Definitions

3.3.1 nbPodState – pod state enumeration

Enumeration: nbPodState	
Enumeration Values:	
Unknown (-1)	Pod state is unknown
Disconnected (0)	Pod is unplugged or offline
Error (1)	Pod has failed
Normal (2)	Pod is operating normally
Description: This enumeration is used to show the operational state of nbPod objects.	

3.3.2 nbEnclosureRelLoc – relative location within enclosure

Enumeration: nbEnclosureRelLoc	
Enumeration Values:	
NA (0)	N/A
Above (1)	Device is located above enclosure
Below (2)	Device is located below enclosure
CenterInterior (3)	Device is located in center of interior of enclosure
TopInterior (4)	Device is located in top of interior of enclosure
BottomInterior (5)	Device is located in bottom of interior of enclosure
LeftPower (6)	Device is located at left power rail for enclosure
RightPower (7)	Device is located at right power rail for enclosure
AirIntake (8)	Device is located at air intake for enclosure
AirExhaust (9)	Device is located at air exhaust for enclosure
FrontDoor (10)	Device is located on front door of enclosure
BackDoor (11)	Device is located on back door of enclosure
AtUPS (12)	Device is located at UPS for enclosure
AtCooling (13)	Device is located at cooling for enclosure
AtPDU (14)	Device is located at power distribution unit for enclosure
Description: This enumeration is used to provide a set of relative location options for device position within a rack or other enclosure.	

3.3.3 nbAlertSeverity – alert severity

Enumeration: nbAlertSeverity	
Enumeration Values:	
Information (0)	Information severity level (lowest)
Warning (1)	Warning severity level
Error (2)	Error severity level (default for thresholds)
Critical (3)	Critical severity level
Failure (4)	Failure severity level (highest, used for unplugged pods and sensors)
Description: This enumeration provides the different alert severity levels.	

3.3.4 nbErrorStatus – error status

Enumeration: nbErrorStatus	
Enumeration Values:	
None (-1)	No alert
Information (0)	Information severity alert
Warning (1)	Warning severity alert
Error (2)	Error severity alert
Critical (3)	Critical severity alert
Failure (4)	Failure severity alert
Description: This enumeration provides the possible error status values, which consist of the same alert severities as the nbAlertSeverity enumeration, plus the “None” value for no error.	

3.3.5 nbPortState – port state enumeration

Enumeration: nbPortState	
Enumeration Values:	
Disconnected (0)	Port is unplugged or offline
Error (1)	Port has failed
Normal (2)	Port is operating normally
Description: This enumeration is used to show the operational state of nbPort objects.	

3.3.6 nbUnitsRequest – request preferred units system

Enumeration: nbUnitsRequest	
Enumeration Values:	
Default (0)	Use default units system for sensor values
English (1)	Use English units for sensor values, if possible
Metric (2)	Use Metric units for sensor values, if possible
Description: This enumeration is used to select which units system should be used for sensor values and other units-sensitive data.	

3.3.7 nbScheduleTime – possible times for a schedule to be active

Enumeration: nbScheduleTime	
Enumeration Values:	
At0000 (0)	12:00am
At0015 (1)	12:15am
At0030 (2)	12:30am
At0045 (3)	12:45am
At0100 (4)	1:00am
At0115 (5)	1:15am
At0130 (6)	1:30am
At0145 (7)	1:45am
At0200 (8)	2:00am
At0215 (9)	2:15am
At0230 (10)	2:30am
At0245 (11)	2:45am
At0300 (12)	3:00am
At0315 (13)	3:15am
At0330 (14)	3:30am
At0345 (15)	3:45am
At0400 (16)	4:00am
At0415 (17)	4:15am
At0430 (18)	4:30am
At0445 (19)	4:45am
At0500 (20)	5:00am
At0515 (21)	5:15am
At0530 (22)	5:30am
At0545 (23)	5:45am
At0600 (24)	6:00am
At0615 (25)	6:15am

At0630 (26)	6:30am
At0645 (27)	6:45am
At0700 (28)	7:00am
At0715 (29)	7:15am
At0730 (30)	7:30am
At0745 (31)	7:45am
At0800 (32)	8:00am
At0815 (33)	8:15am
At0830 (34)	8:30am
At0845 (35)	8:45am
At0900 (36)	9:00am
At0915 (37)	9:15am
At0930 (38)	9:30am
At0945 (39)	9:45am
At1000 (40)	10:00am
At1015 (41)	10:15am
At1030 (42)	10:30am
At1045 (43)	10:45am
At1100 (44)	11:00am
At1115 (45)	11:15am
At1130 (46)	11:30am
At1145 (47)	11:45am
At1200 (48)	12:00pm
At1215 (49)	12:15pm
At1230 (50)	12:30pm
At1245 (51)	12:45pm
At1300 (52)	1:00pm
At1315 (53)	1:15pm
At1330 (54)	1:30pm
At1345 (55)	1:45pm
At1400 (56)	2:00pm
At1415 (57)	2:15pm
At1430 (58)	2:30pm
At1445 (59)	2:45pm
At1500 (60)	3:00pm
At1515 (61)	3:15pm
At1530 (62)	3:30pm
At1545 (63)	3:45pm
At1600 (64)	4:00pm
At1615 (65)	4:15pm
At1630 (66)	4:30pm
At1645 (67)	4:45pm

At1700 (68)	5:00pm
At1715 (69)	5:15pm
At1730 (70)	5:30pm
At1745 (71)	5:45pm
At1800 (72)	6:00pm
At1815 (73)	6:15pm
At1830 (74)	6:30pm
At1845 (75)	6:45pm
At1900 (76)	7:00pm
At1915 (77)	7:15pm
At1930 (78)	7:30pm
At1945 (79)	7:45pm
At2000 (80)	8:00pm
At2015 (81)	8:15pm
At2030 (82)	8:30pm
At2045 (83)	8:45pm
At2100 (84)	9:00pm
At2115 (85)	9:15pm
At2130 (86)	9:30pm
At2145 (87)	9:45pm
At2200 (88)	10:00pm
At2215 (89)	10:15pm
At2230 (90)	10:30pm
At2245 (91)	10:45pm
At2300 (82)	11:00pm
At2315 (93)	11:15pm
At2330 (94)	11:30pm
At2345 (95)	11:45pm
Description: This enumeration provides all the possible times in a given day that scheduled activities can be controlled for (on a 15 minute boundary). As an index, the time period for each is simply the value times 15 minutes after midnight.	

3.3.8 nbAlertStateFilter – select which alert states to return

Enumeration: nbAlertStateFilter	
Enumeration Values:	
Any (0)	Both active and resolved alerts
Active (1)	Only active alerts
Resolved (2)	Only resolved alerts
Description: This enumeration is used to determine if alerts in the resolved, active, or any state should be returned.	

3.3.9 nbAlertAckRC – result codes for acknowledgeAlert() method

Enumeration: nbAlertAckRC	
Enumeration Values:	
Success (0)	Alert was resolved successfully
NoAckNeeded (1)	Alert was already resolved
AccessDenied (2)	Not allowed to update alert
GeneralError (3)	Function failed
Description: This enumeration defines the result codes for the acknowledgeAlert() method.	

3.3.10 nbCameraColorBalance – standard color balance options

Enumeration: nbCameraColorBalance	
Enumeration Values:	
AutoColorBalance (0)	Automatic color balance
CustomColorBalance (1)	Customer set color balance
Fluorescent (2)	Fluorescent light color balance preset
Incandescent (3)	Incandescent light color balance preset
Daylight (4)	Daylight color balance preset
Description: This enumeration defines the standard color balance options.	

3.3.11 nbThresholdSensorType – type of sensor associated with a threshold

Enumeration: nbThresholdSensorType	
Enumeration Values:	
NumSensor (0)	Numeric sensor (nbNumSensor)
StateSensor (1)	State sensor (nbStateSensor)
StringSensor (2)	String sensor (nbStringSensor)
Description: This enumeration indicates which type of sensor a threshold is associated with.	

3.3.12 nbThresholdParmType – indicates the data type for a threshold parameter

Enumeration: nbThresholdParmType	
Enumeration Values:	
FloatingPointValue (0)	Floating point value (double)
IntegerValue (1)	Integer value (int)
StringValue (2)	String value (string)
BooleanValue (3)	Boolean value (bool)
ScheduleValue (4)	Schedule value (nbSchedule)
StringListValue (5)	String list value (string[])
Description: This enumeration is used to indicate what general data type is used for a given parameter in a threshold.	

3.3.13 nbThresholdUpdateRC – result codes for threshold update methods

Enumeration: nbThresholdUpdateRC	
Enumeration Values:	
Success (0)	Threshold was updated successfully
AccessDenied (1)	Not allowed to update threshold
ObjectDoesntExist (2)	Threshold does not exist
GeneralError (3)	Function failed
Description: This enumeration defines the result codes for the updateThreshold() and deleteThreshold() methods.	

3.3.14 nbAlertProfileScheduleCaptureOpt – options for forcing data captures during alert profile schedule execution

Enumeration: nbAlertProfileScheduleCaptureOpt	
Enumeration Values:	
CaptureWhenNeeded (0)	Capture the given data when one or more of the executed alert actions request it (default)
CaptureAlways (1)	Capture the given data whenever the schedule runs (independent of any actions)
CaptureNever (2)	Never capture the given data, even if alert actions request it
Description: This enumeration defines the choices for driving the capture of graph, picture, and audio resources during the execution of an alert profile.	

3.3.15 nbAlertActionParmType – indicates the data type for an alert action parameter

Enumeration: nbAlertActionParmType	
Enumeration Values:	
FloatingPointValue (0)	Floating point value (double)
IntegerValue (1)	Integer value (int)
StringValue (2)	String value (string)
BooleanValue (3)	Boolean value (bool)
ScheduleValue (4)	Schedule value (nbSchedule)
StringListValue (5)	String list value (string[])
OrderedStringListValue (6)	Ordered string list value (string[])
Description: This enumeration is used to indicate what general data type is used for a given parameter in an alert action.	

3.3.16 nbAlertActionUpdateRC – result codes for alert action update methods

Enumeration: nbAlertActionUpdateRC	
Enumeration Values:	
Success (0)	Alert action was updated successfully
AccessDenied (1)	Not allowed to update alert action
ObjectDoesntExist (2)	Alert action does not exist
GeneralError (3)	Function failed
Description: This enumeration defines the result codes for the updateAlertAction() and deleteAlertAction() methods.	

3.3.17 nbAlertProfileUpdateRC – result codes for alert profile update methods

Enumeration: nbAlertProfileUpdateRC	
Enumeration Values:	
Success (0)	Alert profile was updated successfully
AccessDenied (1)	Not allowed to update alert profile
ObjectDoesntExist (2)	Alert profile does not exist
GeneralError (3)	Function failed
Description: This enumeration defines the result codes for the updateAlertProfile() and deleteAlertProfile() methods.	

3.3.18 nbUserAccountUpdateRC – result codes for user account update methods

Enumeration: nbUserAccountUpdateRC	
Enumeration Values:	
Success (0)	User account was updated successfully
AccessDenied (1)	Not allowed to update user account
ObjectDoesntExist (2)	User account does not exist
GeneralError (3)	Function failed
Description: This enumeration defines the result codes for the updateUserAccount() and deleteUserAccount() methods.	

3.3.19 nbConfigUpdateRC – result codes for configuration update methods

Enumeration: nbConfigUpdateRC	
Enumeration Values:	
Success (0)	Configuration was updated successfully
AccessDenied (1)	Not allowed to update configuration
ObjectDoesntExist (2)	Configuration object does not exist
GeneralError (3)	Function failed
Description: This enumeration defines the result codes for the setEmailServerSettings() method.	

3.3.20 nbUseSSLChoices – configuration choices for SSL connections

Enumeration: nbUseSSLChoices	
Enumeration Values:	
NeverUseSSL (0)	Never attempt to use SSL
UseSSLIfAvailable (1)	Use SSL if available
RequireSSLAnyCert (2)	Use SSL and accept any certificate (including self-signed)
RequireSSLValidCert (3)	Use SSL and only accept trusted certifications
RequireSSLStrictCert (4)	Use SSL and only accept trusted certificates with matching hostnames
Description: This enumeration defines the options for communications using SSL.	

4 Appendices

4.1 WSDL for Alert Receiver Web Service

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://www.netbotz.com/BotzWare"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://www.netbotz.com/BotzWare"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://www.netbotz.com/BotzWare">
      <s:element name="BotzWareNumericSensorAlert">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="Appliance"
type="tns:nbPod" />
            <s:element minOccurs="0" maxOccurs="1" name="Alert"
type="tns:nbAlert" />
            <s:element minOccurs="0" maxOccurs="1" name="Sensor"
type="tns:nbNumSensor" />
            <s:element minOccurs="0" maxOccurs="1"
name="PortForSensor" type="tns:nbPort" />
            <s:element minOccurs="0" maxOccurs="1"
name="PodForSensor" type="tns:nbPod" />
            <s:element minOccurs="1" maxOccurs="1" name="IsResolved"
type="s:boolean" />
            <s:element minOccurs="0" maxOccurs="1" name="UserArg"
type="s:string" />
            <s:element minOccurs="1" maxOccurs="1" name="Units"
type="tns:nbUnitsRequest" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="nbPod">
        <s:complexContent mixed="false">
          <s:extension base="tns:nbObject">
            <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="State"
type="tns:nbPodState" />
              <s:element minOccurs="0" maxOccurs="1" name="Label"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="Parent"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="DockedTo"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="SerialNum"
type="s:string" />
```

```
              <s:element minOccurs="0" maxOccurs="1" name="Location"
type="tns:nbLocationData" />
              <s:element minOccurs="0" maxOccurs="1" name="Product"
type="tns:nbProductData" />
              <s:element minOccurs="1" maxOccurs="1"
name="StartupTime" type="s:dateTime" />
              <s:element minOccurs="0" maxOccurs="1"
name="EncFolderID" type="s:string" />
              <s:element minOccurs="1" maxOccurs="1"
name="UnplugAlertSev" type="tns:nbErrorStatus" />
              <s:element minOccurs="1" maxOccurs="1"
name="ErrorStatus" type="tns:nbErrorStatus" />
              <s:element minOccurs="1" maxOccurs="1"
name="Integrated" type="s:boolean" />
              <s:element minOccurs="0" maxOccurs="1" name="PortID"
type="s:string" />
            </s:sequence>
          </s:extension>
        </s:complexContent>
      </s:complexType>
      <s:complexType name="nbObject">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="ID"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="GUID"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Class"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="ClassPath"
type="tns:ArrayOfString" />
          <s:element minOccurs="1" maxOccurs="1" name="ReadOnly"
type="s:boolean" />
          <s:element minOccurs="1" maxOccurs="1" name="Constant"
type="s:boolean" />
          <s:element minOccurs="1" maxOccurs="1" name="NoDelete"
type="s:boolean" />
          <s:element minOccurs="1" maxOccurs="1" name="Persistent"
type="s:boolean" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="ArrayOfString">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded"
name="string" nillable="true" type="s:string" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="nbPort">
        <s:complexContent mixed="false">
          <s:extension base="tns:nbObject">
            <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="State"
type="tns:nbPortState" />
              <s:element minOccurs="0" maxOccurs="1" name="Label"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="PodID"
type="s:string" />
```

```
            </s:sequence>
          </s:extension>
        </s:complexContent>
      </s:complexType>
      <s:simpleType name="nbPortState">
        <s:restriction base="s:string">
          <s:enumeration value="Disconnected" />
          <s:enumeration value="Error" />
          <s:enumeration value="Normal" />
        </s:restriction>
      </s:simpleType>
      <s:complexType name="nbSensor">
        <s:complexContent mixed="false">
          <s:extension base="tns:nbObject">
            <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="Label"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="PodID"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="Location"
type="tns:nbLocationData" />
              <s:element minOccurs="1" maxOccurs="1"
name="ErrorStatus" type="tns:nbErrorStatus" />
              <s:element minOccurs="0" maxOccurs="1" name="PortID"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1"
name="SuggestedThreshClass" type="s:string" />
              <s:element minOccurs="1" maxOccurs="1"
name="IsMonitored" type="s:boolean" />
            </s:sequence>
          </s:extension>
        </s:complexContent>
      </s:complexType>
      <s:complexType name="nbLocationData">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="Location"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Country"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="State"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="City"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Address1"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Address2"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Company"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Site"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Building"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Floor"
type="s:string" />
```

```
          <s:element minOccurs="0" maxOccurs="1" name="Room"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Latitude"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Longitude"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="RoomRow"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="RoomColumn"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="EnclosureID"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="SlotLocation"
type="s:string" />
          <s:element minOccurs="1" maxOccurs="1"
name="RelativeLocation" type="tns:nbEnclosureRelLoc" />
          <s:element minOccurs="0" maxOccurs="1"
name="HeightAboveFloor" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Contact"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Notes"
type="s:string" />
        </s:sequence>
      </s:complexType>
      <s:simpleType name="nbEnclosureRelLoc">
        <s:restriction base="s:string">
          <s:enumeration value="NA" />
          <s:enumeration value="Above" />
          <s:enumeration value="Below" />
          <s:enumeration value="CenterInterior" />
          <s:enumeration value="TopInterior" />
          <s:enumeration value="BottomInterior" />
          <s:enumeration value="LeftPower" />
          <s:enumeration value="RightPower" />
          <s:enumeration value="AirIntake" />
          <s:enumeration value="AirExhaust" />
          <s:enumeration value="FrontDoor" />
          <s:enumeration value="BackDoor" />
          <s:enumeration value="AtUPS" />
          <s:enumeration value="AtCooling" />
          <s:enumeration value="AtPDU" />
        </s:restriction>
      </s:simpleType>
      <s:simpleType name="nbErrorStatus">
        <s:restriction base="s:string">
          <s:enumeration value="None" />
          <s:enumeration value="Information" />
          <s:enumeration value="Warning" />
          <s:enumeration value="Error" />
          <s:enumeration value="Critical" />
          <s:enumeration value="Failure" />
        </s:restriction>
      </s:simpleType>
      <s:complexType name="nbNumSensor">
        <s:complexContent mixed="false">
          <s:extension base="tns:nbSensor">
```

```
            <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="Value"
type="s:double" />
              <s:element minOccurs="0" maxOccurs="1" name="UnitsID"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1"
name="AltUnitsID" type="s:string" />
              <s:element minOccurs="1" maxOccurs="1" name="MinValue"
type="s:double" />
              <s:element minOccurs="1" maxOccurs="1" name="MaxValue"
type="s:double" />
              <s:element minOccurs="1" maxOccurs="1" name="ValueInc"
type="s:double" />
              <s:element minOccurs="1" maxOccurs="1"
name="HistoryTime" type="s:int" />
              <s:element minOccurs="1" maxOccurs="1"
name="MaxHistoryTime" type="s:int" />
              <s:element minOccurs="0" maxOccurs="1" name="ValueFmt"
type="s:string" />
            </s:sequence>
          </s:extension>
        </s:complexContent>
      </s:complexType>
      <s:complexType name="nbAlert">
        <s:complexContent mixed="false">
          <s:extension base="tns:nbObject">
            <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="Label"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1"
name="Description" type="tns:ArrayOfString" />
              <s:element minOccurs="0" maxOccurs="1"
name="AlertTypeID" type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="PodID"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="SensorID"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="PortID"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="ProfileID"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1"
name="ThresholdID" type="s:string" />
              <s:element minOccurs="1" maxOccurs="1" name="Severity"
type="tns:nbAlertSeverity" />
              <s:element minOccurs="1" maxOccurs="1" name="StartTime"
type="s:dateTime" />
              <s:element minOccurs="1" maxOccurs="1"
name="ResolveTime" type="s:dateTime" />
              <s:element minOccurs="1" maxOccurs="1"
name="IsResolved" type="s:boolean" />
              <s:element minOccurs="0" maxOccurs="1"
name="NotifyList" type="tns:ArrayOfString" />
              <s:element minOccurs="0" maxOccurs="1"
name="CameraList" type="tns:ArrayOfString" />
```

```
              <s:element minOccurs="1" maxOccurs="1"
name="WasCancelled" type="s:boolean" />
              <s:element minOccurs="1" maxOccurs="1" name="Latitude"
type="s:double" />
              <s:element minOccurs="1" maxOccurs="1" name="Longitude"
type="s:double" />
              <s:element minOccurs="0" maxOccurs="1" name="URL"
type="s:string" />
              <s:element minOccurs="0" maxOccurs="1"
name="UserDescription" type="tns:ArrayOfString" />
              <s:element minOccurs="1" maxOccurs="1"
name="IsManualResolve" type="s:boolean" />
              <s:element minOccurs="0" maxOccurs="1"
name="ManualResolveUserID" type="s:string" />
              <s:element minOccurs="0" maxOccurs="1"
name="ManualResolveNote" type="s:string" />
            </s:sequence>
          </s:extension>
        </s:complexContent>
      </s:complexType>
      <s:simpleType name="nbAlertSeverity">
        <s:restriction base="s:string">
          <s:enumeration value="Information" />
          <s:enumeration value="Warning" />
          <s:enumeration value="Error" />
          <s:enumeration value="Critical" />
          <s:enumeration value="Failure" />
        </s:restriction>
      </s:simpleType>
      <s:simpleType name="nbPodState">
        <s:restriction base="s:string">
          <s:enumeration value="Unknown" />
          <s:enumeration value="Disconnected" />
          <s:enumeration value="Error" />
          <s:enumeration value="Normal" />
        </s:restriction>
      </s:simpleType>
      <s:complexType name="nbProductData">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="Vendor"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Type"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Model"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="FullModel"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="SubModel"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="SerialNum"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Manufacturer"
type="s:string" />
          <s:element minOccurs="1" maxOccurs="1" name="ManufDate"
type="s:dateTime" />
```

```
          <s:element minOccurs="0" maxOccurs="1" name="Revision"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="BoardID"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="BootVersion"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="AppVersion"
type="s:string" />
        </s:sequence>
      </s:complexType>
      <s:simpleType name="nbUnitsRequest">
        <s:restriction base="s:string">
          <s:enumeration value="Default" />
          <s:enumeration value="English" />
          <s:enumeration value="Metric" />
        </s:restriction>
      </s:simpleType>
      <s:element name="BotzWareNumericSensorAlertResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1"
name="BotzWareNumericSensorAlertResult" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="BotzWareStateSensorAlert">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="Appliance"
type="tns:nbPod" />
            <s:element minOccurs="0" maxOccurs="1" name="Alert"
type="tns:nbAlert" />
            <s:element minOccurs="0" maxOccurs="1" name="Sensor"
type="tns:nbStateSensor" />
            <s:element minOccurs="0" maxOccurs="1"
name="PortForSensor" type="tns:nbPort" />
            <s:element minOccurs="0" maxOccurs="1"
name="PodForSensor" type="tns:nbPod" />
            <s:element minOccurs="1" maxOccurs="1" name="IsResolved"
type="s:boolean" />
            <s:element minOccurs="0" maxOccurs="1" name="UserArg"
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="nbStateSensor">
        <s:complexContent mixed="false">
          <s:extension base="tns:nbSensor">
            <s:sequence>
              <s:element minOccurs="1" maxOccurs="1"
name="ValueIndex" type="s:int" />
              <s:element minOccurs="0" maxOccurs="1" name="ValueEnum"
type="tns:ArrayOfString" />
              <s:element minOccurs="1" maxOccurs="1"
name="HistoryTime" type="s:int" />
```

```
              <s:element minOccurs="1" maxOccurs="1"
name="MaxHistoryTime" type="s:int" />
            </s:sequence>
          </s:extension>
        </s:complexContent>
      </s:complexType>
      <s:element name="BotzWareStateSensorAlertResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1"
name="BotzWareStateSensorAlertResult" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="BotzWareStringSensorAlert">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="Appliance"
type="tns:nbPod" />
            <s:element minOccurs="0" maxOccurs="1" name="Alert"
type="tns:nbAlert" />
            <s:element minOccurs="0" maxOccurs="1" name="Sensor"
type="tns:nbStringSensor" />
            <s:element minOccurs="0" maxOccurs="1"
name="PortForSensor" type="tns:nbPort" />
            <s:element minOccurs="0" maxOccurs="1"
name="PodForSensor" type="tns:nbPod" />
            <s:element minOccurs="1" maxOccurs="1" name="IsResolved"
type="s:boolean" />
            <s:element minOccurs="0" maxOccurs="1" name="UserArg"
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="nbStringSensor">
        <s:complexContent mixed="false">
          <s:extension base="tns:nbSensor">
            <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="Value"
type="s:string" />
            </s:sequence>
          </s:extension>
        </s:complexContent>
      </s:complexType>
      <s:element name="BotzWareStringSensorAlertResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1"
name="BotzWareStringSensorAlertResult" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="BotzWarePortAlert">
        <s:complexType>
          <s:sequence>
```

```
            <s:element minOccurs="0" maxOccurs="1" name="Appliance"
type="tns:nbPod" />
            <s:element minOccurs="0" maxOccurs="1" name="Alert"
type="tns:nbAlert" />
            <s:element minOccurs="0" maxOccurs="1" name="Port"
type="tns:nbPort" />
            <s:element minOccurs="0" maxOccurs="1" name="PodForPort"
type="tns:nbPod" />
            <s:element minOccurs="1" maxOccurs="1" name="IsResolved"
type="s:boolean" />
            <s:element minOccurs="0" maxOccurs="1" name="UserArg"
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="BotzWarePortAlertResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1"
name="BotzWarePortAlertResult" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="BotzWarePodAlert">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="Appliance"
type="tns:nbPod" />
            <s:element minOccurs="0" maxOccurs="1" name="Alert"
type="tns:nbAlert" />
            <s:element minOccurs="0" maxOccurs="1" name="Pod"
type="tns:nbPod" />
            <s:element minOccurs="1" maxOccurs="1" name="IsResolved"
type="s:boolean" />
            <s:element minOccurs="0" maxOccurs="1" name="UserArg"
type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="BotzWarePodAlertResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1"
name="BotzWarePodAlertResult" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="BotzWareNumericSensorAlertSoapIn">
    <wsdl:part name="parameters"
element="tns:BotzWareNumericSensorAlert" />
  </wsdl:message>
  <wsdl:message name="BotzWareNumericSensorAlertSoapOut">
    <wsdl:part name="parameters"
element="tns:BotzWareNumericSensorAlertResponse" />
```

```
  </wsdl:message>
  <wsdl:message name="BotzWareStateSensorAlertSoapIn">
    <wsdl:part name="parameters"
element="tns:BotzWareStateSensorAlert" />
  </wsdl:message>
  <wsdl:message name="BotzWareStateSensorAlertSoapOut">
    <wsdl:part name="parameters"
element="tns:BotzWareStateSensorAlertResponse" />
  </wsdl:message>
  <wsdl:message name="BotzWareStringSensorAlertSoapIn">
    <wsdl:part name="parameters"
element="tns:BotzWareStringSensorAlert" />
  </wsdl:message>
  <wsdl:message name="BotzWareStringSensorAlertSoapOut">
    <wsdl:part name="parameters"
element="tns:BotzWareStringSensorAlertResponse" />
  </wsdl:message>
  <wsdl:message name="BotzWarePortAlertSoapIn">
    <wsdl:part name="parameters" element="tns:BotzWarePortAlert" />
  </wsdl:message>
  <wsdl:message name="BotzWarePortAlertSoapOut">
    <wsdl:part name="parameters"
element="tns:BotzWarePortAlertResponse" />
  </wsdl:message>
  <wsdl:message name="BotzWarePodAlertSoapIn">
    <wsdl:part name="parameters" element="tns:BotzWarePodAlert" />
  </wsdl:message>
  <wsdl:message name="BotzWarePodAlertSoapOut">
    <wsdl:part name="parameters"
element="tns:BotzWarePodAlertResponse" />
  </wsdl:message>
  <wsdl:portType name="nbAlertReceiverServiceSoap">
    <wsdl:operation name="BotzWareNumericSensorAlert">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Alert
receiver interface for sending numeric sensor alerts.</documentation>
      <wsdl:input message="tns:BotzWareNumericSensorAlertSoapIn" />
      <wsdl:output message="tns:BotzWareNumericSensorAlertSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="BotzWareStateSensorAlert">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Alert
receiver interface for sending state sensor alerts.</documentation>
      <wsdl:input message="tns:BotzWareStateSensorAlertSoapIn" />
      <wsdl:output message="tns:BotzWareStateSensorAlertSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="BotzWareStringSensorAlert">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Alert
receiver interface for sending string sensor alerts.</documentation>
      <wsdl:input message="tns:BotzWareStringSensorAlertSoapIn" />
      <wsdl:output message="tns:BotzWareStringSensorAlertSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="BotzWarePortAlert">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Alert
receiver interface for sending port alerts.</documentation>
      <wsdl:input message="tns:BotzWarePortAlertSoapIn" />
      <wsdl:output message="tns:BotzWarePortAlertSoapOut" />
    </wsdl:operation>
```

```
    <wsdl:operation name="BotzWarePodAlert">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Alert
receiver interface for sending pod alerts.</documentation>
      <wsdl:input message="tns:BotzWarePodAlertSoapIn" />
      <wsdl:output message="tns:BotzWarePodAlertSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="nbAlertReceiverServiceSoap"
type="tns:nbAlertReceiverServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
    <wsdl:operation name="BotzWareNumericSensorAlert">
      <soap:operation
soapAction="http://www.netbotz.com/BotzWare/BotzWareNumericSensorAler
t" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="BotzWareStateSensorAlert">
      <soap:operation
soapAction="http://www.netbotz.com/BotzWare/BotzWareStateSensorAlert"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="BotzWareStringSensorAlert">
      <soap:operation
soapAction="http://www.netbotz.com/BotzWare/BotzWareStringSensorAlert
" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="BotzWarePortAlert">
      <soap:operation
soapAction="http://www.netbotz.com/BotzWare/BotzWarePortAlert"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="BotzWarePodAlert">
```

```
      <soap:operation
soapAction="http://www.netbotz.com/BotzWare/BotzWarePodAlert"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="nbAlertReceiverService">
    <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Sample
NetBotz Alert Receiver Web Service</documentation>
    <wsdl:port name="nbAlertReceiverServiceSoap"
binding="tns:nbAlertReceiverServiceSoap">
      <soap:address
location="http://localhost/nbSampleAlertReceiver/nbSampleAlertReceive
r.asmx" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

APC Worldwide Customer Support

Customer support for this or any other APC product is available at no charge in any of the following ways:

- Visit the APC Web site to access documents in the APC Knowledge Base and to submit customer support requests.
  - **www.apc.com** (Corporate Headquarters)
    Connect to localized APC Web sites for specific countries, each of which provides customer support information.
  - **www.apc.com/support/**
    Global support searching APC Knowledge Base and using e-support.
- Contact the APC Customer Support Center by telephone or e-mail.
  - Local, country-specific centers: go to **www.apc.com/support/contact** for contact information.

For information on how to obtain local customer support, contact the APC representative or other distributors from whom you purchased your APC product.

990-3538

1/2009