CHIPKIN
Toll Free: 1 866 383-1657
3381 Cambie St., #211 Vancouver, BC Canada V5Z 4R3

Chipkin™ BACnet Command line interface
USER MANUAL

VERSION: 0.1.0
PRODUCT NUMBER: CAS-2300-18
DOCUMENT REVISION: 1

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

COPYRIGHT & TRADEMARKS

DISCLAIMER

Chipkin Automation Systems™ has limited its liability for damages incurred by the customer or its personnel in the contractual documents pursuant to which the product is provided to the customer. The information and specifications contained throughout this user manual are up to date at the time of publication. Chipkin Automation Systems has used, and continues to use, its best efforts to maintain this user manual to reflect the most current configuration of the product. Chipkin Automation Systems reserves the right to change the contents of this user manual at any time without notice and assumes no liability for its accuracy. In the preparation of this user manual, Chipkin Automation Systems has incorporated, and/or compiled service information and maintenance procedures sourced from manufacturers and vendors of parts and components used in the manufacturing of this product. Therefore, Chipkin Automation Systems shall not be liable for omissions or missing data. It is not the intension of this user manual to instruct service technicians in using common sense, basic skills and
rules of service repair.

CONFIDENTIALITY

The information contained in this document is the intellectual property of Chipkin Automation Systems and is Commercially Confidential. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Chipkin Automation Systems.

DOCUMENT REVISION HISTORY

REVISION	DATE	AUTHOR	NOTE
1	2018-Sep-21	SWS	Created

Table 1 - Document Revision History

1. PREFACE

1.1 WELCOME

As a new owner of Chipkin Automation Systems'™ (CAS) Gateway you have joined thousands of satisfied customers who use Chipkin's protocol gateways, data clients, application, and integration services to meet their building and industrial automation requirements. Our configuration expertise in this field combined with free BACnet and other tools ensure your success; and our customer support via phone, email and remote desktop tools means that we're there when you need us. Thank you for choosing Chipkin's products.

1.2 CHIPKIN

Chipkin offers expert solutions for your building and industrial automation requirements. We develop, configure, install and support gateways (protocol converters), data loggers, and remote monitor and controlling applications. Founded in October 2000, Chipkin provides expert solutions for converting BACnet®, Modbus®, and Lonworks®—to name just a few—and enabling interfaces for HVAC, fire, siren, intercom, lighting, transportation and fuel systems. The high-quality products we offer (including those from other vendors) interface with Simplex™, Notifier™, McQuay™, GE™ and many others—so you can rest assured that Chipkin will select the most appropriate solution for your application.

1.3 CUSTOMER SUPPORT

Chipkin is a small responsive company, and we live or die by the quality of our service—and with offices in two time-zones—we can provide support when you need it. For information on sales, service, obtaining documentation or submitting a service request, please call us toll free at 1-866-383-1657. Thanks for choosing Chipkin's protocol gateways, data clients and integration services to meet your building and industrial automation requirements.

SALES AND CUSTOMER SUPPORT
TOLL FREE: 1-866-383-1657
FAX: 1-416-915-4024
EMAIL: salesgroup1@chipkin.com

GENERAL
TOLL FREE: 1-866-383-1657
FAX: 1-416-915-4024
EMAIL: support@chipkin.com

SHIPPING ADDRESS
3381 Cambie St., #211
Vancouver, BC,
Canada V5Z 4R3

2. OVERVIEW

2.1 PRODUCT SUMMARY

Chipkin BACnet Command line (CAS BACnet Cli) interface is a command line application that sends BACnet client messages and parses BACnet response from BACnet devices as human readable text or as machine readable XML.

The CAS BACnet Cli application can be used for many purposes including;

- Create scriptable end to end integration testing for BACnet server applications.
- Extending existing application to discover, integrate, write devices, objects and properties of BACnet devices.
- Testing BACnet server applications.

BACNET BIBB

Supported BACnet BIBB

- **DS-RP-A:** Data Sharing - ReadProperty-A
- **DS-RPM-A:** Data Sharing - ReadPropertyMultiple-A
- **DS-WP-A:** Data Sharing - WriteProperty-A
- **DS-WPM-A:** Data Sharing - WritePropertyMultiple-A
- **DS-COV-A:** Data Sharing - Change of Value-A
- **DS-COVP-A:** Data Sharing - Change of Value Property-A
- **DS-COVU-A:** Data Sharing - Change of Value Unsubscribed-A
- **DM-DDB-A:** Device and Network Management - Dynamic Device Binding-A
- **DM-DOB-A:** Device and Network Management - Dynamic Object Binding-A
- **DM-OCD-A:** Device Management - Object Creation and Deletion-A
- **DM-UTC-A:** Device Management - UTCTimeSynchronization-A
- **DM-TS-A:** Device Management - TimeSynchronization-A
- **NM-FDR-A:** Foreign Device Registration-A

3. CONNECTIONS

3.1 NETWORK CONNECTIONS

This block diagram lists common network connections that are used with the CAS BACnet Cli application.

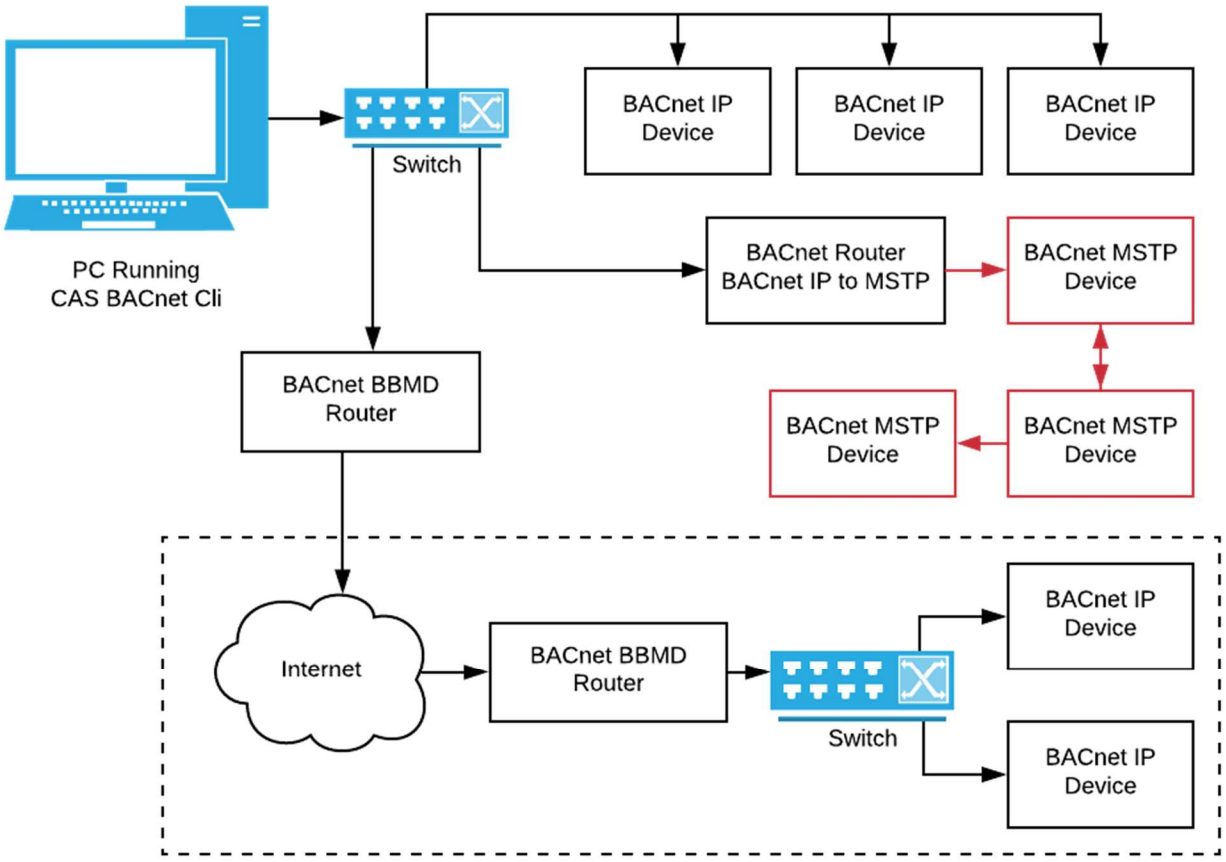

Figure 3.1-1. Network Connections Block Diagram

3.2 COMUNICATION PORTS

The CAS BACnet Cli application uses the following ports for communication over the ethernet port.

Protocol	Port	Notes
BACnet	UDP 47808	BACnet protocol, can be configured

Table 2 - Communication Ports

4. COMMANDS

The CAS BACnet Cli application supports the following commands

- **Who-Is** - Discover BACnet devices on the local network.
- **Who-Has** - Discover BACnet devices on the local network by object name or instance number.
- **Read Property** - Get the current value of an object's property.
- **Write Property** - Set the current value of an object's property.
- **Subscribe COV** - Subscribe to changes of value for an object's property.
- **Cancel Subscribe COV** - Unsubscribe to changes of value for an object's property.
- **Register Foreign Device** - Registers a device with a BBMD.
- **Create Object** - Creates an object on a device.
- **Delete Object** - Delete an object on a device.
- **I-Am** - Sends a I-AM message.
- **Build Write Property** - Stores a combination of Object identifier, Property Identifier, and value to disk to be used by the **Send Built Write Property** command.
- **Send Built Write Property** - Sends the stored combinations created by the *Build Write Property* command as a Write Property Multiple message.
- **Build Read Property** - Stores a combination of Object identifier, and Property Identifier to disk to be used by the *Send Built Read Property* command.
- **Send Built Read Property** - Sends the stored combinations created by the *Build Read Property* command as a Read Property Multiple message.
- **Build Create Object** - Stores a combination of Object identifier, Property Identifier, and value to disk to be used by *Send Built Create Object*
- **Send Built Create Object** - Sends the stored combinations created by the *Build Create Object* command as a Create Object message.
- **Reset** - Removes temporary files created by whois, whohas, buildcreateobject, buildwriteproperty, buildreadproperty
- **Enum** - Lists the enumerations used by the CASBACnetCli command line parameters

Common command line parameters

- **-v –verbose** - Print debug information in human readable format
- **-t <time>, --timeout=<time>** - The APDU timeout in seconds [default: 3]
- **--port=<port>** - Destination udp port [default: 47808]
- **--ip=<ipaddress>** - Destination ip address.
- **--dnet=<destination-network>** - Destination network on foreign network.
- **--dadr=<destination-address>** - Destination address (in HEXon foreign network.
- **--clientDeviceID=<deviceID>** - The command line tool's device ID. [default: 3899999]

4.1 Who-is

Sends a Who-is message onto the network. This message is used to determine the device object identifier, the network address, vendors ID, and MAX APDU accepted. The Who-is message can optionally be restricted only discover devices with a device object identifier within a certain range using the [<low-limit>] [<high-limit>] optional parameters.

Note:

- If <*low-limit*> is defined but <*high-limit*> is not defined, then <*high-limit*> will equal to <*low-limit*>

Usage:

```
CASBACnetCli whois [<low-limit>] [<high-limit>] [options]
```

Arguments:

- **<low-limit>** - [Optional] Device Instance Range Low Limit. This parameter is an unsigned integer in the range 0 - 4194303. In conjunction with the 'Device Instance Range High Limit' parameter, it defines the devices that are qualified to respond with an I-Am service request. Device Object_Identifier instance number falls within the range greater than or equal to 'Device Instance Range Low Limit' and less than or equal to 'Device Instance Range High Limit' shall be qualified to respond.
- **<high-limit>** - [Optional] Device Instance Range High Limit. This parameter is an unsigned integer in the range 0 - 4194303. In conjunction with the 'Device Instance Range Low Limit' parameter, it defines the devices that are qualified to respond with an I-Am service request. See <low-limit> for more information.

Examples:

Sends a whois discovery message on the network for all devices regardless of their device object identifier

```
CASBACnetCli whois
```

Sends a whois discovery message on the network for a device with device object instance of 389000. This is equivalent to calling "CASBACnetCli whois 389000 389000"

```
CASBACnetCli whois 389000
```

Sends a ranged whois discovery message on the network looking for devices with device instances between 1000 and 2000

```
CASBACnetCli whois 1000 2000
```

4.2 Who-Has

Sends a Who-has message onto the network. This message is used to determine the device object identifier, the network address, vendors ID, and MAX APDU accepted given Object_Name or a given Object_Identifier. A I-Have message is sent in response by a device that matches the search criteria in a Who-has message.

Note:

- For a list of the enumerations for <object-type> run "CASBACnetCli enum object-type"

Usage:

```
CASBACnetCli whohas <object-name> [options]
CASBACnetCli whohas <object-type> <object-instance> [options]
```

Arguments:

- **<object-name>** - The 'Object Name' parameter, of type CharacterString, shall convey the value of the Object_Name property of the object that is to be located. If the 'Object Name' parameter is present, then only those devices that contain an object with an Object_Name property value matching the 'Object Name' parameter shall respond with an I-Have response.
- **<object-type>** - Object type is part of the The 'Object Identifier' parameter, of type BACnetObjectIdentifier, shall convey the Object_Identifier of the object that is to be located. If the 'Object Identifier' parameter is present, then only those devices that contain an object with an Object_Identifier property value matching the 'Object Identifier' parameter shall respond with an I-Have response.
- **<object-instance>** - Object instance is part of the The 'Object Identifier' parameter of type BACnetObjectIdentifier. See <object-type> for more information.

Examples:

Search for devices with objects that have the Object name of "Current temperature"

```
CASBACnetCli whohas "Current temperature"
```

Search for a device (8) with a instance number of 389001

```
CASBACnetCli whohas 8 389001
```

Search for a device with a analog input (0) with with a instance number of 801

```
CASBACnetCli whohas 0 801
```

4.3 Read Property

Sends a read property request for a current value of a object's property. This service allows read access to any property of any object, whether a BACnet-defined object or not. If <property-identifier> parameter is one of the special property identifiers ALL (8), REQUIRED (105), or OPTIONAL (80) then the service will be ReadPropertyMultiple instead of ReadProperty

Note:

- For a list of the enumerations for <object-type> run "CASBACnetCli enum object-type"
- For a list of the enumerations for <property-identifier> run "CASBACnetCli enum property-identifier"

Usage:

```
CASBACnetCli readproperty <object-type> <object-instance> <property-identifier> [<property-array-index>] [options]
```

Arguments:

- **<object-type>** - Object type combined with Object instance is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<object-instance>** - Object instance combined with Object type is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<property-identifier>** - This parameter, of type BACnetPropertyIdentifier, shall provide the means of uniquely identifying the property to be read and returned by this service.
- **<property-array-index>** - [Optional] If the property identified above is of datatype array, this optional parameter of type Unsigned shall indicate the array index of the element of the property referenced by this service. If the 'Property Array Index' is omitted, this shall mean that the entire array shall be referenced.

Examples:

Sends a read property request for the object-name (77) from a device (8) with a instance of 389000

```
CASBACnetCli readproperty 8 389000 77
```

Sends a read property request for the object-list (76) with a array index 19 from a device (8) with a instance of 389000.

```
CASBACnetCli readproperty 8 389000 76 19
```

Sends a read property request for the present-Value (85) from a analog input (0) with a instance of 800

```
CASBACnetCli readproperty 0 800 85
```

4.4 Write Property

The WriteProperty service is used by a client BACnet-user to modify the value of a specified property of a BACnet object. This service potentially allows write access to any property of any object, whether a BACnet-defined object or not. Some implementers may wish to restrict write access to certain properties of certain objects. In such cases, an attempt to modify a restricted property shall result in the return of an error of 'Error Class' PROPERTY and 'Error Code' WRITE_ACCESS_DENIED.

Note:

- For a list of the enumerations for <object-type> run `"CASBACnetCli enum object-type"`
- For a list of the enumerations for <property-identifier> run `"CASBACnetCli enum property-identifier"`
- For a list of the enumerations for <datatype> run `"CASBACnetCli enum datatype"`

Usage:

```
CASBACnetCli writeproperty <object-type> <object-instance> <property-identifier> <value> [--datatype=<datatype>] [--priority=<priority-level>] [options]
```

Arguments:

- **<object-type>** - Object type combined with Object instance is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<object-instance>** - Object instance combined with Object type is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.

- **<property-identifier>** - This parameter, of type BACnetPropertyIdentifier, shall provide the means of uniquely identifying the property to be written by this service. Because this service is intended to write a single property of a single object, the value of this parameter shall not be one of the special property identifiers ALL (8), REQUIRED (105), or, OPTIONAL(80).
- **<value>** - If access to the specified property of the specified object is successful, this parameter shall be used to replace the value of the property at the time of access. It shall be of any datatype that is valid for the property being modified. If the datatype parameter is not defined then the datatype will attempt to be discovered.
- **--datatype=<datatype>** - [Optional] Defines the data type of the <value> parameter.
- **--priority=<priority-level>** - [Optional] This parameter shall be an integer in the range 1-16, which indicates the priority assigned to this write operation. If an attempt is made to write to a commandable property without specifying the priority, a default priority of 16 (the lowest priority) shall be assumed. If an attempt is made to write to a property that is not commandable with a specified priority, the priority shall be ignored.

Examples:
Sends a write property message to update the present value (85) to 99.6 of a analogInput (0) with a object instance of 800. The datatype is assumed to be Real (4) and the priorty is defaulted to the lowest value of 16.

```
CASBACnetCli writeproperty 0 800 85 99.6
```

Sends a write property message to update the present value (85) to 99.6 of a analogInput (0) with a object instance of 800. The datatype is set to Real (4) and the priorty is defaulted to the lowest value of 16.

```
CASBACnetCli writeproperty 0 800 85 99.6 --datatype=4
```

Sends a write property message to update the present value (85) to 99.6 of a analogInput (0) with a object instance of 800. The datatype is set to Real (4) and the priorty is set to 8

```
CASBACnetCli writeproperty 0 800 85 99.6 --datatype=4 --priority=8
```

Sends a write property message to update the object name (77) to "new name" of a device (8) with a object instance of 389000. The datatype is set to CharacterString (4)

```
CASBACnetCli writeproperty 8 389000 77 "new name" --datatype=7
```

4.5 Subscribe COV

Sends a SubscribeCOV message to subscribe for the receipt of notifications of changes that may occur to the properties of a particular object. Certain BACnet standard objects may optionally support COV reporting. If a standard object provides COV reporting, then changes of value of specific properties of the object, in some cases based on programmable increments, trigger COV notifications to be sent to one or more subscriber clients.

Note:

- For a list of the enumerations for <object-type> run `"CASBACnetCli enum object-type"`

Usage:

```
CASBACnetCli subscribecov <object-type> <object-instance> <lifetime>
<subscriber-process-identifier> [--enable-confirmed-notifications]
[options]
```

Arguments:

- **<object-type>** - Object type combined with Object instance is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<object-instance>** - Object instance combined with Object type is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<lifetime>** - This parameter, of type Unsigned, shall convey the desired lifetime of the subscription in seconds. A value of zero shall indicate an indefinite lifetime, without automatic cancellation. A non-zero value shall indicate the number of seconds that may elapse before the subscription shall be automatically cancelled.
- **<subscriber-process-identifier>** - This parameter, of type Unsigned32, shall convey a numeric "handle" meaningful to the subscriber. This handle shall be used to match future re-subscriptions and cancellations from the subscriber with the COV context that exists within the COVserver device and with confirmed or unconfirmed COV notifications to identify the process within the COV-client that should receive them.
- **--enable-confirmed-notifications** - [Optional] This parameter is a flag, shall convey whether the COV-server device shall issue ConfirmedCOVNotifications (TRUE) or UnconfirmedCOVNotifications (FALSE) when changes occur.

Examples:

Send a Subscribe COV message for a Analog input (0) with a object instance of 800 and a lifetime of 300 seconds (5 mins). The subscriber process identifier is 101.

```
CASBACnetCli subscribecov 0 800 300 101
```

Send a Subscribe COV message for a Analog input (0) with a object instance of 800 and a lifetime of 300 seconds (5 mins). The subscriber process identifier is 101. The responding device uses a ConfirmedCOVNotifications instead of UnconfirmedCOVNotifications.

```
CASBACnetCli subscribecov 0 800 300 101 --enable-confirmed-
notifications
```

4.6 Cancel Subscribe COV

Send a Cancel Subscribe COV message to cancel a existing subscription. The <object-type> <object-instance> and <subscriber-process-identifier> of the previous subscription is required.

Usage:

```
CASBACnetCli cancelsubscribecov <object-type> <object-instance>
<subscriber-process-identifier> [options]
```

Note:

- For a list of the enumerations for <object-type> run "`CASBACnetCli enum object-type`"

Arguments:

- **<object-type>** - Object type combined with Object instance is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<object-instance>** - Object instance combined with Object type is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<subscriber-process-identifier>** - This parameter, of type Unsigned32, shall convey a numeric "handle" meaningful to the subscriber. This handle shall be used to match future re-subscriptions and cancellations from the subscriber with the COV context that exists within the COV server device and with confirmed or unconfirmed COV notifications to identify the process within the COV-client that should receive them.

Examples:

Send a Cancel Subscribe COV message to cancel a existing subscription for a Analog input (0) with a object instance of 800 and a subscriber process identifier of 101.

```
CASBACnetCli cancelsubscribecov 0 800 101
```

4.7 Register Foreign Device

XXXX Description XXXX TODO

Usage:

CASBACnetCli registerforeigndevice <lifetime> [options]

Arguments:

- **<lifetime>** - This parameter, of type Unsigned, shall convey the desired lifetime in seconds of the foreign registration.

Examples:

Sends a foreign device registration message with a lifetime of 300 seconds (5 mins) to a specific BACnet BBMD gateway at IP 192.168.1.25

```
CASBACnetCli registerforeigndevice 300 --ip=192.168.1.26
```

4.8 Create Object

Sends a Create Object message to create a new instance of an object. This service may be used to create instances of both standard and vendor specific objects.

Note:

- For a list of the enumerations for <object-type> run `"CASBACnetCli enum object-type"`

Usage:

```
CASBACnetCli createobject <object-type> [<object-instance>] [options]
```

Arguments:

- **<object-type>** - Object type combined with Object instance is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<object-instance>** - [Optional] Object instance combined with Object type is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.

Examples:

Sends a create object message asking a remote device to create a Analog input (0) object.

```
CASBACnetCli createobject 0
```

Sends a create object message asking a remote device to create a Analog input (0) object with a object instance of 800

```
CASBACnetCli createobject 0 800
```

4.9 Delete Object

Sends a DeleteObject message to a device to delete an existing object. Although this service is general in the sense that it can be applied to any object type, it is expected that most objects in a control system cannot be deleted by this service because they are protected as a security feature. There are some objects, however, that may be created and deleted dynamically. Group objects and Event Enrollment objects are examples.

Note:

- For a list of the enumerations for <object-type> run "CASBACnetCli enum object-type"

Usage:

```
CASBACnetCli deleteobject <object-type> <object-instance> [options]
```

Arguments:

- **<object-type>** - Object type combined with Object instance is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<object-instance>** - Object instance combined with Object type is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.

Examples:

Sends a delete object message to a device in an attempt to delete an analog input (0) with a object instance of 800

```
CASBACnetCli deleteobject 0 800
```

4.10 I-Am

Sends an IAM message for the CASBACnetCli

Usage:

```
CASBACnetCli iam [options]
```

Examples:
Sends a I-am message onto the local network.

```
CASBACnetCli iam
```

4.11 Build Write Property and Send Built Write Property

The buildwriteproperty command records to a file, combination of BACnetObjectIdentifier, BACnetPropertyIdentifier and value. This command does not send any messages onto the network.

The sendbuiltwriteproperty command is used to send the recorded combination created by buildwriteproperty as a single WritePropertyMuliple message. After this command is successfully executed the recorded combination created by buildwriteproperty are cleared.

The reset command clears the recorded combination created by buildwriteproperty command.

Note:

- For a list of the enumerations for <object-type> run "CASBACnetCli enum object-type"
- For a list of the enumerations for <property-identifier> run "CASBACnetCli enum property-identifier"

Usage:

```
CASBACnetCli buildwriteproperty <object-type> <object-instance>
<property-identifier> <value> [--priority=<priority-level>] [--
datatype=<datatype>] [options]

CASBACnetCli sendbuiltwriteproperty [options]

CASBACnetCli reset
```

Arguments:

- **<object-type>** - Object type combined with Object instance is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<object-instance>** - Object instance combined with Object type is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.

- **<property-identifier>** - This parameter, of type BACnetPropertyIdentifier, shall provide the means of uniquely identifying the property to be written to.
- **<value>** - This parameter in combination with the <property-identifier>, shall be used to set the value of this object.
- **--priority=<priority-level>**
- **--datatype=<datatype>** - [Optional] Defines the data type of the <value> parameter.

Examples:
The following series of commands build a series of combination of BACnetObjectIdentifier,BACnetPropertyIdentifier, and values. In this case a write request for object-name (77), present-value (85) on a analog-output (1) with a object instance of 800. Then sends the write property multiple message with the previous series of combination of BACnetObjectIdentifier BACnetPropertyIdentifier, and value as a single message.

```
CASBACnetCli buildwriteproperty 1 800 77 "New analog output name" --
datatype=7
CASBACnetCli buildwriteproperty 1 800 85 99.6 --datatype=4 --
priority=8
CASBACnetCli sendbuiltwriteproperty
```

4.12 Build Read Property and Send Built Read Property

The buildreadproperty command records to a file, combination of BACnetObjectIdentifier and BACnetPropertyIdentifier. This command does not send any messages onto the network.

The sendbuiltreadproperty command is used to send the recorded combination created by buildreadproperty as a single ReadPropertyMuliple message. After this command is successfully executed the recorded combination created by buildreadproperty are cleared.

The reset command clears the recorded combination created by buildreadproperty command.

Note:
- For a list of the enumerations for <object-type> run `"CASBACnetCli enum object-type"`
- For a list of the enumerations for <property-identifier> run `"CASBACnetCli enum property-identifier"`

Usage:

```
CASBACnetCli buildreadproperty <object-type> <object-instance>
<property-identifier> [<property-array-index>] [options]

CASBACnetCli sendbuiltreadproperty [options]

CASBACnetCli reset
```

Arguments:

- **<object-type>** - Object type combined with Object instance is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<object-instance>** - Object instance combined with Object type is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<property-identifier>** - This parameter, of type BACnetPropertyIdentifier, shall provide the means of uniquely identifying the property to be read and returned by this service.
- **<property-array-index>** - [Optional] If the property identified above is of datatype array, this optional parameter of type Unsigned shall indicate the array index of the element of the property referenced by this service. If the 'Property Array Index' is omitted, this shall mean that the entire array shall be referenced.

Examples:

The following series of commands build a series of combination of BACnetObjectIdentifier and BACnetPropertyIdentifier. In this case a request for object-name (77), status-flags (111), description (28) on a device (8) with a object instance of 389000. Then sends the Read property Multiple message with the previous series of combination of BACnetObjectIdentifier and BACnetPropertyIdentifier as a single message.

```
CASBACnetCli buildreadproperty 8 389000 77
CASBACnetCli buildreadproperty 8 389000 111
CASBACnetCli buildreadproperty 8 389000 28
CASBACnetCli sendbuiltreadproperty
```

4.13 Build Create Object and Send Built Create Object

The buildcreateobject command records to a file, combination of BACnetPropertyIdentifier and value. This command does not send any messages onto the network.

The sendbuiltcreateobject command is used to send the recorded combination created by buildcreateobject as a single CreateObjectmessage. After this command is successfully executed the recorded combination created by buildcreateobject are cleared.

The reset command clears the recorded combination created by buildcreateobject command.

Note:

- For a list of the enumerations for <object-type> run "CASBACnetCli enum object-type"
- For a list of the enumerations for <property-identifier> run "CASBACnetCli enum property-identifier"
- For a list of the enumerations for <datatype> run "CASBACnetCli enum datatype"

Usage:

```
CASBACnetCli buildcreateobject <property-identifier> <value> [--
datatype=<datatype>]

CASBACnetCli sendbuiltcreateobject <object-type> [<object-instance>]
[options]

CASBACnetCli reset
```

Arguments:

- **<object-type>** - Object type combined with Object instance is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<object-instance>** - Object instance combined with Object type is part of the 'Object Identifier' parameter, of type BACnetObjectIdentifier.
- **<property-identifier>** - This parameter, of type BACnetPropertyIdentifier, shall provide the means of uniquely identifying the property to be initialize on creation of this object.
- **<value> -** This parameter in combination with the <property-identifier>, shall be used to initialize properties on the creation of this object.
- **--datatype=<datatype> -** [Optional] Defines the data type of the <value> parameter.

Examples:

The following series of commands build a series of combination of BACnetPropertyIdentifier, and values. In this case a on creation of the analog-output (1) with a object instance of 800 The intial value of object-name (77), and present-value (85) wil be set.

```
CASBACnetCli buildcreateobject 77 "New analog output name" --
datatype=7
CASBACnetCli buildcreateobject 85 99.6 --datatype=4 --priority=8
```

```
CASBACnetCli sendbuiltcreateobject 1 800
```

4.14 Enum

Prints the enumerated values for the parameters used by the CASBACnetCli application.

Usage:

```
CASBACnetCli enum
CASBACnetCli enum <enum-name>
CASBACnetCli enum <enum-name> <enum-value>
```

Arguments:

- **<enum-name>** - [Optional] The name of the enumeration to print. Options include object-type, property-identifier, datatype
- **<enum-value>** - [Optional] The value of a defined enumeration to print. This is useful for finding a single enumeration in a large list.

Examples:

Prints all enumerations

```
CASBACnetCli enum
```

Prints the object type enumerations

```
CASBACnetCli enum object-type
```

Prints the object type enumeration with a value of 14 (multiStateOutput)

```
CASBACnetCli enum object-type 14
```

Prints all the property-identifier enumerations

```
CASBACnetCli enum property-identifier
```

Prints the object type enumeration with a value of 85 (presentValue)

```
CASBACnetCli enum property-identifier 85
```

APPENDIX A: LIMITED WARRANTY

LIMITED WARRANTY

Chipkin Automation Systems provides a 30-Day Return Window (see Return of Non-Defective Products below) and the following limited warranty. This limited warranty extends only to the original purchaser.

Please note that any warranty services or questions must be accompanied by the order number from the transaction through which the warranted product was purchased. ***The order number serves as your warranty number and must be retained*.** Chipkin Automation Systems will offer no warranty service without this number.

Chipkin Automation Systems warrants this product and its parts against defects in materials or workmanship for *three years labor and one year parts* from the original ship date. During this period, Chipkin Automation Systems will repair or replace defective parts with new or reconditioned parts at Chipkin Automations Systems option, without charge to you. Shipping fees incurred from returns for under-warranty service in the first 30-days will be paid by Chipkin Automation Systems. All shipping fees both to and from Chipkin Automation Systems following this 30-day period must be paid by the customer. All returns, both during and following the 30-day period, must be affected via the Procedures for Obtaining Warranty Service described below.

All original parts (parts installed by Chipkin Automation Systems at the original system build) replaced by Chipkin Automation Systems or its authorized service center, become the property of Chipkin Automation Systems. Any after-market additions or modifications will not be warranted. The gateway system owner is responsible for the payment, at current rates, for any service or repair outside the scope of this limited warranty.

Chipkin Automation Systems makes no other warranty, either express or implied, including but not limited to implied warranties of merchantability, fitness for a particular purpose, or conformity to any representation or description, with respect to this computer other than as set forth below. Chipkin Automation Systems makes no warranty or representation, either express or implied, with respect to any other manufacturer's product or documentation, its quality, performance, merchantability, fitness for a particular purpose, or conformity to any representation or description.

Except as provided below, Chipkin Automation Systems is not liable for any loss, cost, expense, inconvenience or damage that may result from use or inability to use the gateway. Under no circumstances shall Chipkin Automation Systems be liable for any loss, cost, expense, inconvenience or damage exceeding the purchase price of the gateway.

The warranty and remedies set forth below are exclusive and in lieu of all others, oral or written, expressed or implied. No reseller, agent or employee is authorized to make any modification, extension or addition to this warranty.

WARRANTY CONDITIONS

The above Limited Warranty is subject to the following conditions:

1. This warranty extends only to products distributed and/or sold by Chipkin Automation Systems. It is effective only if the products are purchased and operated in Canada or the USA. (Within the USA including US 48 States, Alaska and Hawaii.)
2. This warranty covers only normal use of the gateway. Chipkin Automation Systems shall not be liable under this warranty if any damage or defect results from (i) misuse, abuse, neglect, improper shipping or installation; (ii) disasters such as fire, flood, lightning or improper electric current; or (iii) service or alteration by anyone other than an authorized Chipkin Automation Systems' representative; (iv) damages incurred through irresponsible use, including those resulting from viruses or spyware, overclocking, or other non-recommended practices.
3. You must retain your bill of sale or other proof of purchase to receive warranty service.
4. No warranty extension will be granted for any replacement part(s) furnished to the purchaser in fulfillment of this warranty.
5. Chipkin Automation Systems and its Authorized Service Center accepts no responsibility for any software programs, data or information stored on any media or any parts of any products returned for repair to Chipkin Automation Systems.
6. All pre-installed software programs are licensed to customers under non-Chipkin Automation Systems software vendor's term and conditions provided with the packages.
7. This warranty does not cover any third party software or virus related problems.
8. Chipkin Automation Systems makes no warranty either expressed or implied regarding third-party (non-Chipkin Automation System) software.
9. Thirty-day Return Window does not include opened software, parts, special order merchandise and shipping and handling fees.

RETURN OF NON-DEFECTIVE PRODUCTS

A non-defective product may be returned to Chipkin Automation Systems within thirty (30) days of the invoice date for a refund of the original purchase price with the following amendments/fees:

1. Chipkin Automation Systems will refund neither the original shipping cost nor the shipping and handling fees incurred from the products return. If the original purchase was made under a "Free Shipping" promotion, then a standard $40 fee will be deducted from any return in counter to that offer.

2. No refund will be granted for software which has been opened, used, or tampered with in any way which jeopardized Chipkin Automation Systems ability to remarket or resell the product. Chipkin Automation Systems maintains full discretion in decisions regarding a products fitness for return.
3. Any non-defective returns are subject to a 15% restocking fee, which percentage is taken from the final purchase price less any shipping or handling charges.
4. Quantity purchases of five systems or more are not eligible for return.

To return a defective product, please contact our Customer Service Department for a Return Merchandise Authorization (RMA) number and follow the Return of Products Instructions below. The RMA is valid for 10 days from date of issuance. **Returns will not be accepted without an RMA**. Manufacturer restrictions do apply. Any item missing the UPC on the original packaging may not be returned.

PROCEDURES FOR OBTAINING WARRANTY SERVICE

RMA (Returning Merchandise Authorization) Policy:
If repairs are required, the customer must obtain an RMA number and provide proof of purchase. RMA and services are rendered by Chipkin Automation Systems only. Any shipping costs after 30 days (starting from the original date of purchase) on any item returned for repair is the customers' responsibility. All returned parts must have an
RMA number written clearly on the outside of the package along with a letter detailing the problems and a copy of the original proof of purchase. No COD packages will be accepted. No package will be accepted without an RMA number written on the outside of the package. RMA numbers are only valid for 30 days from the date of issue.

Should you have any problems with your gateway, please follow these procedures to obtain the service:

1. If you have purchased our on-site warranty, please find your warranty# (the order number from the transaction through which the warranted product was originally purchased) and contact Chipkin Automation Systems Customer Service by phone at 1-866-383-1657 (Toll free) or 1-647-557-3330.
2. If the gateway must be repaired, an RMA number (Return Merchandise Authorization Number) will be issued for shipment to our repair department. Please follow the instructions given by Chipkin Automation Systems technical support staff to ship your gateway. Chipkin Automation Systems will not accept any shipments without an RMA number.
3. Pack the gateway in its original box or a well-protected box, as outlined in the Return Shipping Instructions. Chipkin Automation Systems will not be responsible for shipping damage/loss of any product outside the original 30-day Chipkin Automation Systems-paid

service period. It is very important that you write the RMA number clearly on the outside of the package. Ship the gateway with a copy of your bill of sale or other proof of purchase, your name, address, phone number, description of the problem(s), and the RMA number you have obtained to:

Chipkin Automation Systems RMA#_____
3381 Cambie St., #211
Vancouver, B.C. Canada, V5Z 4R3

4. Upon receiving the gateway, Chipkin Automation Systems will repair or replace your gateway (at Chipkin Automation Systems discretion) and will ship it back to you within 2 weeks (dependent on parts availability) via UPS.
5. Cross-exchange (Parts only): You will need to provide a valid credit card number as a deposit guarantee when the RMA number is issued. Once approval has been obtained on your credit card, the part(s) will be shipped UPS. You will need to ship defective part(s) back to Chipkin Automation Systems within 15 days to avoid charges to your credit card. If such charges are incurred, the shipped part(s) will be billed at the then current price.
6. Chipkin Automation Systems will pay for shipping to and from the customer only within the first thirty days following the original product ship date. Following this 30-day period all shipping fees both for under warranty and post warranty repairs are the sole responsibility of the customer. The customer also assumes full liability for losses or damages resulting from shipping as well as all responsibility to pursue remuneration for such issues with their selected carrier.

AFTER ONE-YEAR WARRANTY – POST WARRANTY REPAIR

For post warranty repair, the procedure is the same as outlined above for RMA and shipping. However, you are responsible for shipping charges both ways, current labor ($75 per hour if not under warranty), and the current price of part(s) used in repair.

THANK YOU

Thanks for choosing Chipkin's protocol gateways, data clients and integration services to meet your building and industrial automation requirements!

Chipkin Automation Systems™ (Chipkin) is a building and industrial automation protocol expert. We develop, configure, install and support gateways (protocol converters), data loggers and remote monitor and controlling applications.

Founded in October 2000, Chipkin provides expert solutions for converting BACnet®, Modbus®, and Lonworks®—to name just a few—and enabling interfaces for HVAC, fire, siren, intercom, lighting, transportation and fuel systems. The high-quality products we offer (including those from other vendors) interface with Simplex™, Notifier™, McQuay™, GE™ and many others—so you can rest assured that we will select the most appropriate solution for your application.

With Chipkin you are buying a solution. Our configuration expertise in this field combined with free BACnet tools and other tools ensure your success; and our customer support via phone, email and remote desktop tools means that we're there when you need us. Chipkin is a small responsive company, and we live or die by the quality of our service—and with offices in two-time zones—we can provide support when you need it. Give us a call now!

Sales and Customer Service

Toll free: 1-866-383-1657

Email: salesgroup1@chipkin.com

3381 Cambie Street, #211
Vancouver, BC, Canada, V5Z 4R3
1-866-383-1657

Chipkin.com