EUROICC d.o.o.
Tršćanska 21
11080 Zemun - Srbija

PIB: RS100182160
Matični broj: 17196294
Šifra delatnosti: 2651
UniCredit Bank: 170-201-90

Tel/Fax:+38111.371.36.65 ; +38111.371.36.66 e-mail: info@euroicc.com

RingView BACnet User Manual

Document created: 09/10/2012
Status : working version

EUROICC
Tršćanska 21
Belgrade, Serbia
Phone: +381 11 3713 665
Fax: +381 11 3713 666
e-mail: info@euroicc.com

Table of contents

1 Introduction

In “RingBus on BACnet” system, several concepts from these two technologies are merged:

- RBCPU2.1 becomes virtual BACnet device, with b/ip port.
- RingBus elements become BACnet objects. Details of each element-object mapping are in the next chapter.
- RBDIS will be upgraded with special version of RingView application, which will have appropriate tools for editing and saving these mappings, and for sending this configuration to RBCPU2.1's.

1.1 Services Supported

- All special cases of these services are supported (for example: magic numbers for Who-Is, PROP_ALL for Read-Property-Multiple, etc.)
- Who-Is
- Who-Has
- Read-Property
- Read-Property-Multiple
- Subscribe-COV, only for objects which hold error statuses for dampers and FIO modules
- Subscribe-COV-Property, only for objects which hold error statuses for dampers and FIO modules, and only for Present-Value property

1.2 Language

The only character encoding that will be supported is ASCII, so German letters with umlauts will be written in the new form, such as ue, oe, etc.

1.3 BACnet standard objects used in RingBus system

Binary input:

Property identifier	Property datatype	Conformance code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Present_Value	BACnetBinaryPV	R

Status_Flags	BACnetStatusFlags	R
Event_State	BACnetEventState	R
Out_Of_Service	BOOLEAN	R
Polarity	BACnetPolarity	R

Multi-state input:

Property identifier	Property datatype	Conformance code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Present_Value	Unsigned	R
Status_Flags	BACnetStatusFlags	R
Event_State	BACnetEventState	R
Out_Of_Service	BOOLEAN	R
Number_Of_States	Unsigned	R

Analog input:

Property identifier	Property datatype	Conformance code
Object_Identifier	BACnetObjectIdentifier	R
Object_Name	CharacterString	R
Object_Type	BACnetObjectType	R
Present_Value	REAL	R
Status_Flags	BACnetStatusFlags	R
Event_State	BACnetEventState	R
Out_Of_Service	BOOLEAN	R
Units	BACnetEngineeringUnits	R

2 BACnet Objects Tab

Interface for editing BACnet objects can be opened from *System* view clicking on tab *BACnet Objects* as it is shown on next image.

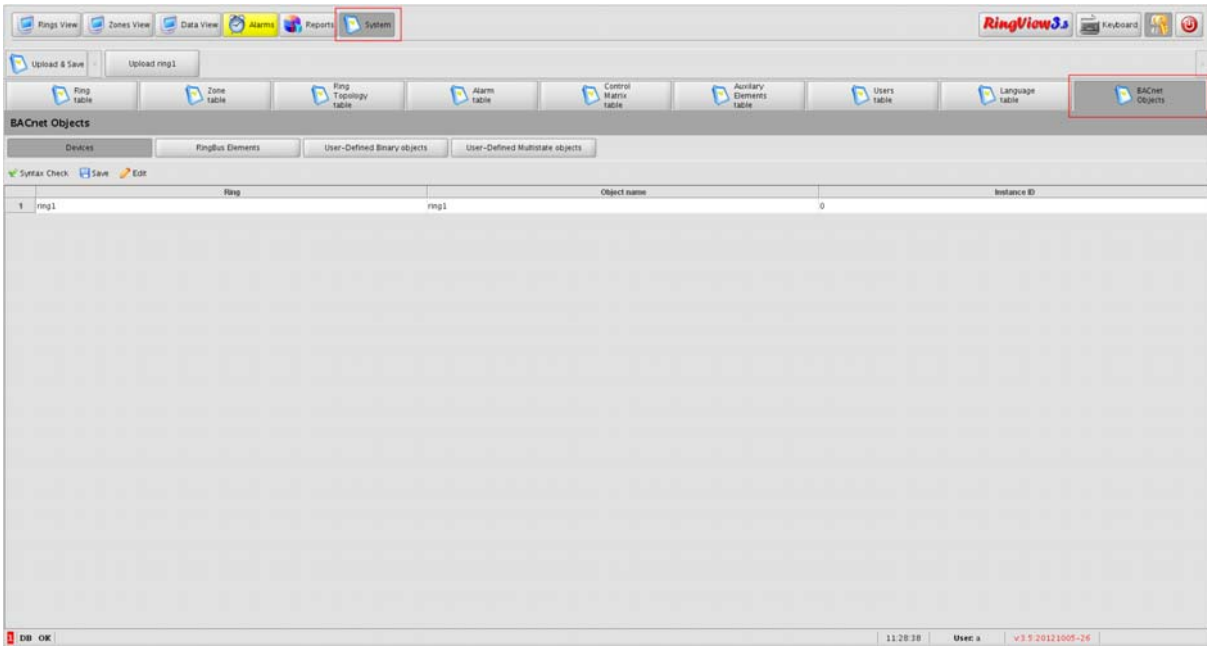

Image 1: BACnet objects tab

When editing BACnet objects following rules must be applied:

1. Object name – Object name must be unique within one device e.g. device and all objects defined on it must have different names.
2. Instance ID
   - Instance ID must be unique within one object type. For example: two binary objects on the same ring with same instance IDs are not allowed.
   - Instance ID must be a number between 0 and 0x3FFFFFF.

In BACnet tab there are four buttons each one for opening table for editing each BACnet object type.

2.1 Device objects

First button shows table for editing device objects as it is shown on next image.

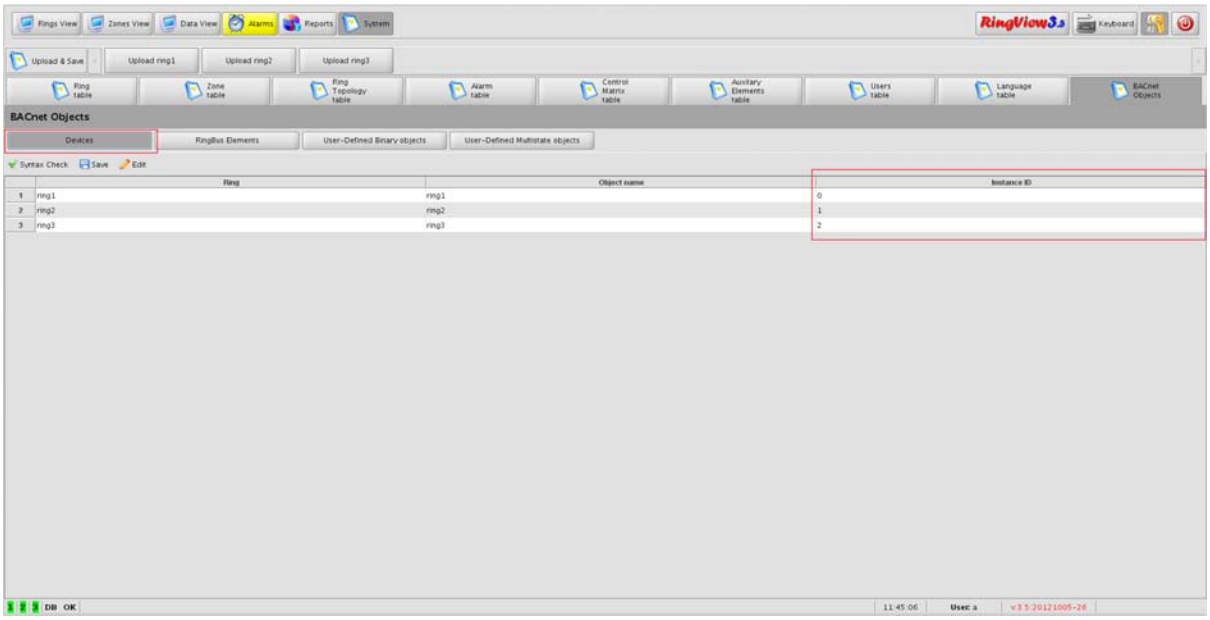

Image 2: Table for editing device objects

Each device object has automatically created object name which is same as ring name. Instance ID is automatically created if it is not defined in *bacnet_devices.csv* file. In this table only Instance ID can be edited. Instance is a number between 0 and 0x3FFFFFF. Toolbar has three buttons:

- *Syntax Check* – Check for errors in this table
- *Save* – Save BACnet objects in corresponding CSV files
- *Edit* – Start editing of selected cell in table if it is enabled for editing. In this table that are only cells in Instance ID column.

2.2 RingBus Elements Objects

Second button show table for editing RingBus Elements Objects which is shown on next image.

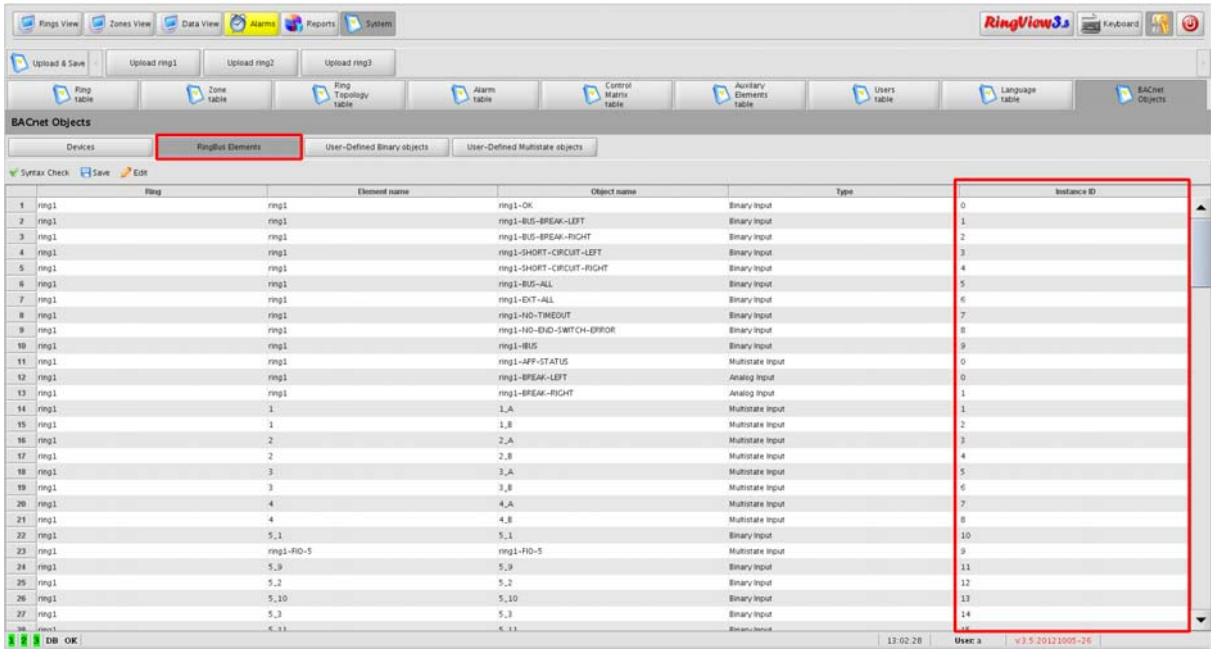

Image 3: Table for editing RingBus elements objects

2.2.1 Element-object mapping

Each element has automatically created object name according following rules.

- **Fire and Smoke dampers**

There are two multi-state input objects for each damper. The first object contains the status messages of each damper. The second object contains error messages of each damper.

- **Multi-state input (status):**

The Multi-state input object name is made of the element name in the matrix, followed by "_B" for status, for example "L01_BSK_15_B".

Name DE	Name EN	Priority
Undefiniert	Irregular state	1
Offen	Open	2
Geschlossen	Close	3
Faehrt auf	Motor ON	4
Faehrt zu	Motor OFF	5

Description field for damper status is copied verbatim from Ring Topology table.

- **Multi-state input (error messages):**

The multi-state input object name is made of the element name in the matrix, followed by "_A" for alarm, for example "L01_BSK_15_A".

Name DE	Name EN	Priority
Servicetaste	Service Pin	1
Laufzeitfehler	Timeout Error	2
Thermokontakt Ausgeloest	Thermo-contact Error	3
Fehler Endschalter	Endswitch Error	4
Fehler Versorgungsspannung	External Power	5
Fehler Busspannung	Bus Power	6
Normal	Normal	7

- **FIO module**

Each digital input and digital output is defined as a separate Binary input object. The

object receives its name from the element in the matrix. Per each FIO module is a multi-state input with error information. The name of the object is automatically generated, but it is also editable by user. For example: “Ring_6_FIO_8”. In this example Ring_6 is name of ring and 8 is unit index of FIO module.

Name DE	Name EN	Priority
Stoerung Kommunikation	Comm Error	1
Fehler Versorgungsspannung	External Power	2
Fehler Busspannung	Bus Power	3
Normal	Normal	4

- **Digital Inputs and Outputs**

Each digital input and digital output is defined as a separate Binary input object. The object receives its name and description from the element in the matrix.

- **System Status**

Each system status will have corresponding Binary, Multi-state and Analog input object, with the name taken from the matrix, in the form “RingName-Status”. Description is automatically generated.

- RingName-OK – True if no errors were found on the ring.
- RingName-BUS-BREAK-LEFT – False if bus is broken from the left, true otherwise.
- RingName-BUS-BREAK-RIGHT – False if bus is broken from the right, true otherwise.
- RingName-SHORT-CIRCUIT-LEFT – False if short circuit is detected from the left, true otherwise.
- RingName-SHORT-CIRCUIT-RIGHT – False if short circuit is detected from the right, true otherwise.
- RingName-BUS-ALL – True if all units in the ring have bus power.
- RingName-EXT-ALL – True if all units in the ring have external power.
- RingName-NO-TIMEOUT – False if at least one unit is in timeout, true otherwise.
- RingName-NO-END-SWITCH-ERROR – False if at least one unit is in end-switch error, true otherwise.
- RingName-IBUS – False if no communication on the bus was possible, true otherwise.
- RingName-APP-STATUS - A single multi-state input will be used for communicating application status on the device (Run, Stop, Bad Configuration).
- RingName-BREAK-LEFT, RingName-BREAK-RIGHT – Two analog inputs will be used for communicating bus breaking from left and right side of the ring.

First column contains element's ring, second element name, third object name, fourth type of object and fifth instance ID. In this table only Instance ID can be edited. Toolbar has same buttons as in panel for editing device objects with same functionality.

2.3 User-defined Binary objects

Table for editing user-defined binary objects is opening clicking on button *User-defined Binary Objects* as it is shown on next image.

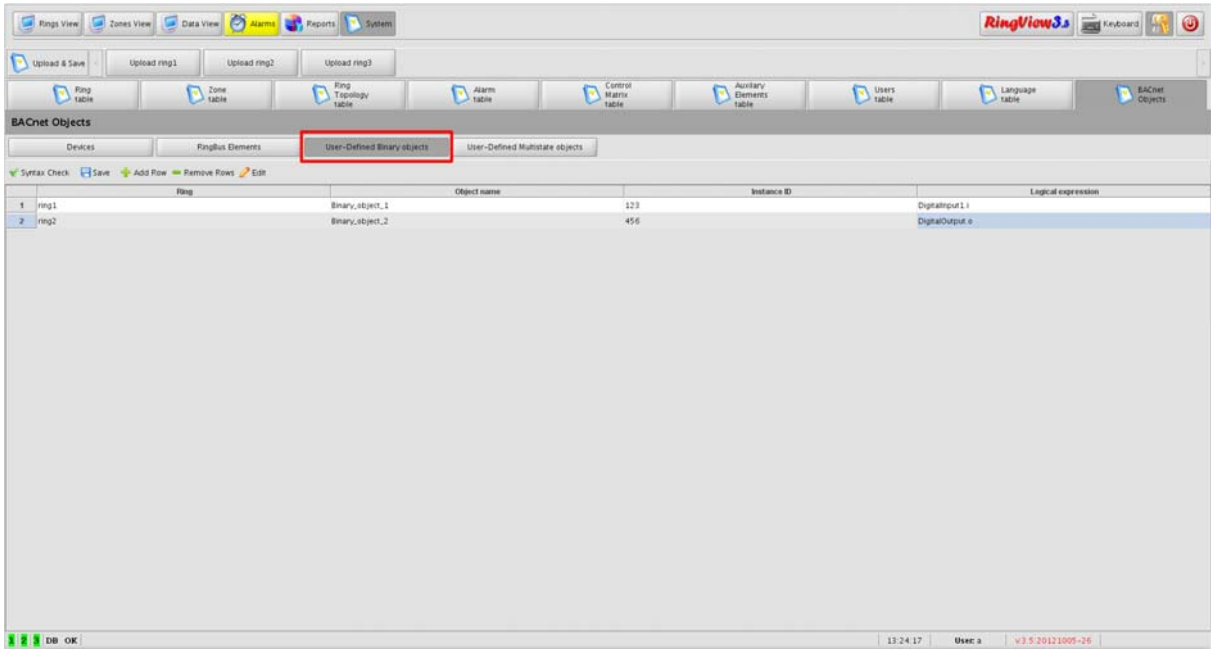

Image 4: Table for editing user-defined binary input objects

In this table all columns are editable. In first column user choose ring from drop-down menu. In the second column is name for binary object. Object name can not contains white spaces and following characters ( ) , . ! | &. Instance ID is setting in third column. Fourth column contains logical expression which determines present value of binary object. Fifth column contains description of the object and it's editable.
Toolbar now contains two new buttons:

- *Add Row* – Adds new row in table
- *Remove Rows* – Removes selected row from table. If there is no selected row, no action is performed.

Other buttons have same functionality as in previous tables.

2.4 User-defined Multistate objects

Table for editing this type of objects is opening by clicking on button User-defined Multistate Objects as it is shown on next image.

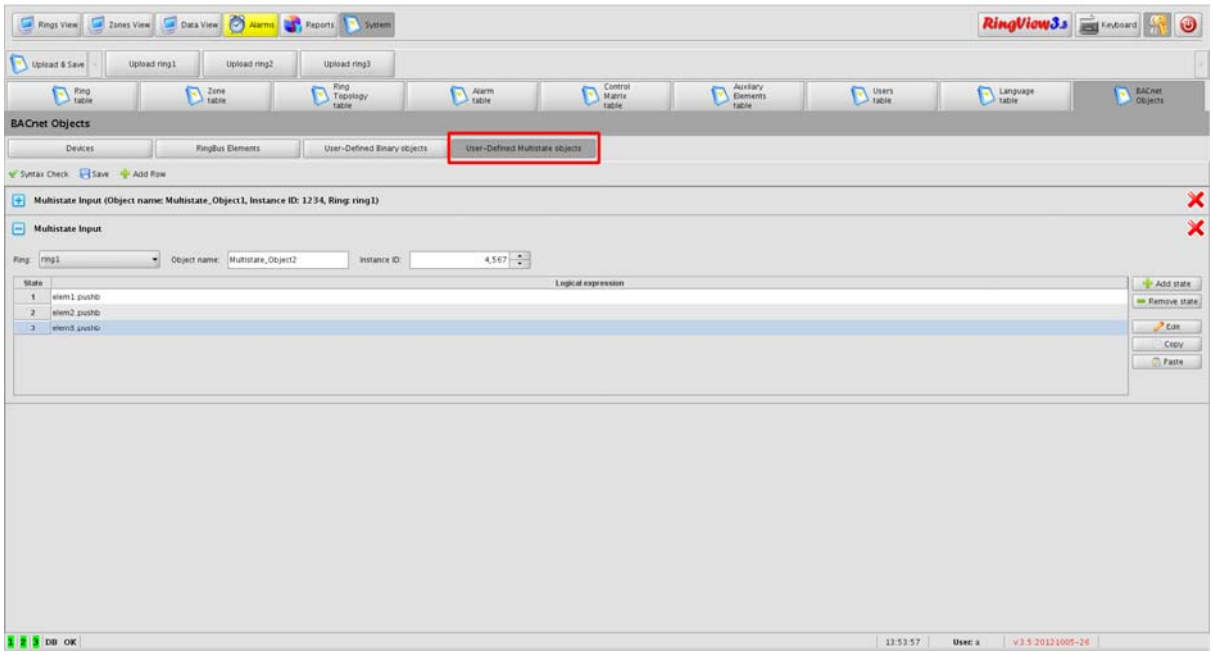

Image 5: Table for editing user-defined multistate input object

On following image is shown detailed information about editing each multistate objects.

Image 6: Detailed view of panel for editing multistate object

Buttons have following functionality:

- *Blue minus button* – Collapse/expand details for multistate object
- *Red X button* – Delete multistate object
- *Add state* – Add new state in states table
- *Remove state* – Remove selected state. If there is no selected state, no action is performed.

- *Edit* – Enable editing selected state in states table
- *Copy* – Copy selected state
- *Paste* – Paste copied state in selected row

Other information for multistate object is:

- Ring which is choosing from drop down menu
- Object name. Object name can not contains white spaces and following characters ( ) , . ! | &.
- Instance ID
- Description of this object, editable by user.
- States – Multistate object must have at least one state. Each state is defined with logical expression.

Toolbar has three buttons with same functionality as described before.

3 CSV files

BACnet objects can be defined via CSV files. If any of this files don't exists, objects will be generated automatically. This applies for CSV files for devices and RingBus elements. Every detected error will be reported and row with error will be skipped. Also missing definitions for devices and RingBus elements will be automatically added. Each object type is defined via one CSV file. That files are:

- *bacnet_rings.csv* – This file contains definitions for rings (devices). Format of this file is:

Index	Ring Name	Instance ID

- *bacnet_elements.csv* – This file contains definitions for RingBus elements. Format of this file is:

Index	Object Name	Type	Ring Name	Instance ID 1	Instance ID 2

Object name is defining according above explained rules in chapter 1.2.1. Type is type of element with special types. For system status sys and for fio element fio.

Instance ID 2 should be defined only for dampers for status object.

- *bacnet_binary_objects.csv* - This file contains definitions for user-defined binary input objects. Format of this file is:

Index	Ring Name	Object Name	Instance ID	Logical expression	Description

- *bacnet_multistate_objects.csv* - This file contains definitions for user-defined multistate input objects. Format of this file is:

Index	Ring Name	Object Name	Instance ID	Description	State 1	…	State N

Every state must be defined in separated column.

4 Terminology

BACnet object – Basic unit of data in BACnet. Common examples are analog/binary/multistate input/output/value, calendar, scheduler and device objects, with many more standard and proprietary types.

BACnet device – Physical or virtual device which serves as root object for accessing other BACnet objects. Each BACnet device holds the list of all objects present on that device.

BACnet network – Network of connected BACnet devices with unique network number. Datalink types for BACnet networks include mstp, arcnet, ethernet, b/ip, and many other proprietary ones.

BACnet router – Physical or virtual device used for interconnecting different BACnet networks.

RingBus element – Physical or virtual device in RingBus system. Each element has several data points, according to its type (fire or smoke dampers, digital inputs and outputs, for example).

RingBus unit – Physical device which can hold arbitrary number of RingBus elements.

RingBus ring – Network of RingBus units, connected to RingBus Modules.

RBCPU2.1 – Main controlling device which collects data from RingBus units and issues commands according to its internal state. Communicates various data with other RBCPU2.1's in the system.

RBDIS – Touchscreen computer with RingView application, used for configuration, monitoring and manual commanding of individual elements. Communicates with RBCPU2.1's.

RingBus System – Consists of at least one RBCPU2.1 with its communication ring. Presence of RBDIS is necessary for configuration of the system; from that moment on, system can work autonomously without user input.

5 Revisions

Rev.	Dat. rev	Description	Author
0.1	09/10/2012	First version	Dragan Bjedov
0.2	10/10/2012	Added introduction and terminology	Dragan Bjedov
0.3	16/10/2012	Added tables for multistate objects Tables for multistate objects changed	Andrija Tomanovic
0.4	6/12/2012	Added COV functionality and description fields	Nikola Jelić
0.5	17/05/2013	Correction: added images	Dragan Bjedov