ADDENDA

BSR/ASHRAE Addendum ba to
ANSI/ASHRAE Standard 135-2012

Data Communication Protocol for Building Automation and Control Networks

Approved by ASHRAE on April 29, 2016, and by the American National Standards Institute on April 29, 2016.

This addendum was approved by a Standing Standard Project Committee (SSPC) for which the Standards Committee has established a documented program for regular publication of addenda or revisions, including procedures for timely, documented, consensus action on requests for change to any part of the standard. The change submittal form, instructions, and deadlines may be obtained in electronic form from the ASHRAE website (www.ashrae.org) or in paper form from the Senior Manager of Standards.

The latest edition of an ASHRAE Standard may be purchased on the ASHRAE website (www.ashrae.org) or from ASHRAE Customer Service, 1791 Tullie Circle, NE, Atlanta, GA 30329-2305. E-mail: orders@ashrae.org. Fax: 678-539-2129. Telephone: 404-636-8400 (worldwide), or toll free 1-800-527-4723 (for orders in US and Canada). For reprint permission, go to www.ashrae.org/permissions.

 ISSN 1041-2336

Approved American National Standard
ANSI
www.ansi.org

[This foreword and the "rationales" on the following pages are not part of this standard. They are merely informative and do not contain requirements necessary for conformance to the standard.]

FOREWORD

The purpose of this addendum is to present changes to ANSI/ASHRAE Standard 135-2012 and Addenda. These modifications are the result of change proposals made pursuant to the ASHRAE continuous maintenance procedures and of deliberations within Standing Standard Project Committee 135. The changes are summarized below.

135-2012*ba*-1. Add CSML Descriptions of BACnet Devices, p. 2
135-2012*ba*-2. Add Semantic Tags to All Objects, p. 15
135-2012*ba*-3. Extend Structured View Object to Contain Semantic Information, p. 16
135-2012*ba*-4. Change Clause 21 identifiers to use a consistent format, p. 19
135-2012*ba*-5. Add Data Revisioning Capabilities to CSML, p. 20

In the following document, language to be added to existing clauses is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are added, plain type is used throughout. All other material in this addendum is provided for context only.

135-2012*ba*-1. Add CSML Descriptions of BACnet Devices.

[This section makes use of additions and changes that are defined in Addendum *am*. Therefore, it is imperative that Addendum 135-2012*am* sections 2, 3, and 6 be applied to the standard before applying this addendum.]

Rationale

This addendum provides a way for BACnet devices to be enhanced to contain a rich set of CSML metadata, definitions, and instances.

This can be used for:

- providing semantic tags on objects and properties;
- providing discoverable arrangements of objects and properties;
- describing proprietary objects, properties, and datatypes;
- providing property metadata like writability, range, volatility, etc.;
- declaring device capabilities (PICS data).

These metadata, definition, and arrangement declarations are deployed as static BACnet File objects or web accessible files, either at the factory and/or by a configuration tool in the field.

[Insert new **Annex X**, p. 1026]

ANNEX X - EXTENDED DISCOVERY OF DEVICES, PROFILES, AND VIEWS (NORMATIVE)

(This annex is part of this standard and is required for its use.)

X.1 Profiles

Every object instance has an optional Profile_Name property. A "profile" describes what a client can expect to find in that object instance, which is especially useful for object types with lots of optional properties, optional writability, or proprietary properties. A profile declaration is specific to an instance; therefore, the value can be different, even for object instances with the same Object_Type.

For example, a device can, and often does, have several kinds of Analog Value objects in a device. Each kind has a particular collection of properties and behavior, but all the BACnet client can see through BACnet services is Object_Type=analog-value, and therefore has no way of knowing what "kind" of Analog Value it is. Consider the following AV kinds, where a Profile_Name would be of value to the client. The first 4 are all "flavors" of the standard Analog Value object with no proprietary properties. The fifth one has some additional proprietary properties that are related to its function as a floating motor controller.

1. Object_Type = analog-value Profile_Name = "555-AV-Status"
2. Object_Type = analog-value Profile_Name = "555-AV-StatusWithAlarm"
3. Object_Type = analog-value Profile_Name = "555-AV-ConfigurationSetting"
4. Object_Type = analog-value Profile_Name = "555-AV-Command"
5. Object_Type = analog-value Profile_Name = "555-AV-FloatingMotor"

While these all have Object_Type=analog-value, by consulting the definitions of the referenced profiles, the client can determine which optional properties are present, what their ranges are, whether they are nonvolatile, etc. In this example, the definition of the last profile indicates to the client that this Analog Value object also has several additional proprietary properties added to it, like "Motor_Full_Travel_Time", which are described in the definition for "555-AV-FloatingMotor". With this information, a user interface client can present descriptions of those proprietary properties to the user for a better interoperability experience.

In order to make efficient use of the definition of a profile, the client must be able to obtain that definition in machine readable form. To enable such automated processing, the definition for a profile is expressed in a file using

the CSML syntax defined in Annex Q. The "Profile_Name" shall be equal to a CSML type name defined in that file. These profile definition files are then placed into zip files that are then referred to as eXtended Data Definition ("xdd") files.

To further aid in automated discovery, the location of an xdd file containing a profile definition can be given with the Profile_Location property. For example,

Profile_Location = "bacnet://.this/file,1"
Profile_Location = "bacnet://6001/file,12"
Profile_Location = "http://somemfg/defs/xdd/AVFM.xdd"
Profile_Location = "http://bms.customer.example.com/xd/dev6001.xdd"

All objects defined in Clause 12 have an optional Profile_Location property. It is recommended that proprietary objects have a Profile_Name property and that devices containing proprietary objects have a Profile_Location property in the Device object. See Clause X.3.

The definition of all profile names with the "0-" prefix is maintained by ASHRAE, in machine readable form, published separately from this standard. It is expected that clients are configured to know the location of this ASHRAE definition file and therefore devices shall not include references to it.

Because Profile_Name is per-instance, not tied to Object_Type, clients can provide support for devices that have "configurable objects," where a programming/configuration tool writes to some proprietary property that changes the property set and behavior of an object instance. When this occurs, the object can properly advertise the new configuration to the clients by updating Profile_Name, and possibly Profile_Location, accordingly.

X.2 xdd Files

The xdd files are containers for a variety of information related to control systems and devices. They can contain a mixture of standardized and non-standardized content. To meet the variety of needs and scenarios described in this standard, they can contain:

- definitions for object profiles and proprietary data structures
- information to define logical arrangements of data
- machine-readable PICS declaration for a device
- human-readable documentation and other helpful and descriptive information about a device
- links to other related xdd files

For example, an xdd file that was discovered from the "Profile_Location" of a BACnet device might contain: the PICS for that device, some links to other xdd files containing common data definitions on the manufacture's website, the data definitions for data specific to that device, information about the arrangement of data within the device, and links to external documentation about that device. Because xdd files can refer to other xdd files, this information might also have been split into multiple xdd files that refer to each other.

X.2.1 xdd File Format

An xdd file is a zip file. The root directory of the zip shall contain the file "ashrae-csml.xml" which provides the main CSML information. Other CSML content can be included for this file. Also, the root directory can optionally contain the file "ashrae-links.txt". All file and directory names in the root directory beginning with "ashrae" are reserved. The presence and content of other files and directories are a local matter. It is recommended, however, that this extra content not be as large as to impact interoperability with consumers. To keep sizes manageable, one xdd file can reference other xdd files. This is accomplished with the links in the "ashrae-links.txt" file.

The format of the "ashrae-csml.xml" file is an XML file, using encoding="utf-8", containing a single <CSML> element, as defined in Annex Q.

Example "ashrae-csml.xml" file (all sections under <CSML> are optional - see Annex Q):

```
<?xml version='1.0' encoding='utf-8'?>
<CSML xmlns="http://bacnet.org/csml/1.2" published="2015-06-29T09:19:44Z" author="..."   ...>
   <Includes ... /> ... see Clause Q.2.1.X2
   <Definitions>
      … definitions for object profiles … see Clause X.3
   </Definitions>
   <Composition name=".pics">
      … PICS information … see Clause X.5
   </Composition >
   ... other possible children of <CSML> ... see Clause Q.2.1
</CSML>
```

The format of the "ashrae-links.txt" file is a plain text file, using UTF-8 (RFC 3629) encoding. The file shall contain a collection of links, each on a separate line, in the format of an HTTP Link Header Field, as defined by Section 5 of RFC 5988. These links are from one xdd file to other xdd files, therefore the links to be considered to be associated with the xdd file resource itself, rather than associated with the inner ashrae-links.txt file. Therefore, relative URI resolution, as defined in Section 5 of REF 3986, is carried out relative to the location of the xdd file.

Example "ashrae-links.txt" file:

```
Link: <http://example.com/xdd/defs/CommonDefs2015E.xdd>; rel="related"; title="Common Definitions"
Link: <http://example.com/xdd/defs/VAVDefs2015B.xdd>; rel="related"; title="VAV Definitions"
```

If an xdd file links to another xdd file in this manner, the client shall consider that the ashrae-csml.xml file in the linked xdd file is implicitly to be included into the ashrae-csml.xml file of the xdd file that contained the link.

The xdd files are expected to be static during normal operation and shall only be changed by a reconfiguration or reprogramming of their corresponding device. It is recommended that clients cache xdd files to avoid unnecessary repeated access and to allow for offline use.

To allow clients to determine when an xdd file has changed, it is required that some externally visible means be provided so that clients do not have to read the file's contents. For http/https accessible files, this can be accomplished by changing the file's name, location, or its HTTP Last-Modified header. For BACnet File objects, this can be accomplished by changing the object's Object_Identifier, Object_Name, or Modification_Date property.

While xdd files can provide beneficial information to clients, the lack of access to any xdd file shall not prevent a client from interacting with standard objects and properties that were defined by this standard for the Protocol_Revision that the client supports.

The registered media type (RFC 6838) for xdd files is "application/bacnet-xdd+zip". HTTP-based clients are advised that not all HTTP servers will correctly set this Content-Type and are recommended to be lenient in the accepted types.

X.2.2 Virtual Objects and Properties

An xdd file can contain virtual instances of BACnet objects. Virtual objects shall be marked with the 'virtual' attribute true. Virtual objects are not included in the Object_List property of the corresponding device. However, the object identifiers and names of virtual objects are nonetheless required to be unique among both virtual and real objects. If a real object with the same object identifier or name is present in the device, then the virtual object is invalidated and shall be ignored. Therefore, creators of xdd files shall take care to ensure that virtual objects and real objects cannot collide.

Clients that are capable of processing xdd files shall be prepared to follow references containing BACnet object identifiers, e.g., BACnetDeviceObjectReference, etc., into the xdd file that is associated with a device via the Device object's Profile_Location and/or Deployed_Profile_Location properties, to find virtual object instances.

As virtual objects can be thought to be logically added to the Device object's Object_List property, virtual Structured View objects that are not included as a subordinate in any other Structured View object for the same device can be thought to be logically added to the Device object's Structured_Object_List property.

X.2.3 Augmentation of Physical Objects

If an object in an xdd file is not marked as virtual, then the data in the xdd file is intended to augment the data in a physical object with the same object identifier and name, usually to add descriptive metadata to the physical property values or to provide virtual properties that are not present in the physical object. Within this object, if a property is not marked as virtual, then the data in the xdd file is intended to augment the value of the physical property. If a property is marked virtual and a real property with the same identifier is present in the device, then the virtual property shall be ignored. Note that augmentation can come from multiple sources. For example, one xdd file could provide 'minimum' and 'maximum' while another provides 'displayName'.

X.3 Example of Definition of Objects, Properties, and Datatypes.

Any vendor's product may contain different "flavors" of standards objects, proprietary extensions to standard objects, or proprietary object types, properties, and datatypes. For maximum interoperability with clients, devices are able to describe this information using an xdd file associated with the device using capabilities provided by the Profile_Name and Profile_Location properties.

Scenario 1: The vendor makes the CSML profile definitions of a proprietary object type available as a static file in the device.

BACnet Proprietary Object:
Object_Identifier= (901,1)
Profile_Name="555-ControlRodsObject"
Profile_Location="bacnet://.this/file,1000"

BACnet File Object:
Object_Identifier = (file,1000)
Contents of zipped file "ashrae-csml.xml" =

```
<?xml version="1.0" encoding="UTF-8"?>
<CSML xmlns=”http://bacnet.org/csml/1.2">
  <Definitions>
    <Object name="555-ControlRodsObject" extends="0-BaseObject" >
      <Real name="command-position" writable="true" commandable="true" units="percent"
        minimum="0.0" maximum="100.0" displayName="Command Position"
        description="The commanded position for the rods - see feedback for actual position"
        propertyIdentifier="1001" tags="control;safety"/>
      <Real name="feedback-position" units="percent" minimum="0.0" maximum="100.0"
        displayName="Actual Position"
        description="The actual position of the rods based on confirmation measurement"
        writableWhen="out-of-service" associatedWith="command-position"
        propertyIdentifier="1002"  tags="feedback;safety"/>
      <!-- This property defines its own proprietary datatype -->
      <Sequence name="safety-limits"   propertyIdentifier="1003" tags="safety;indicatorLevel" >
        <Real name="warn" displayName="Warning!"  contextTag="0" />
        <Real name="high"  displayName="Really!" contextTag="1" />
        <Real name="run"   displayName="Umm...."  contextTag="2"/>
      </Sequence>
      <!-- These properties are linked to each other: you must have at least one. -->
      <Boolean name="horn-enable" optional="true" writable="true" tags="safety;indicator"
        requiredWithout="bell-enable" propertyIdentifier="1007"/>
      <Boolean name="bell-enable" optional="true" writable="true" tags="safety;indicator"
        requiredWithout="horn-enable" propertyIdentifier="1008"/>
    </Object>
    ... possibly other definitions ...
```

```
    <Definitions>
</CSML>
```

Scenario 2: The vendor decides to collect several definitions into a single xdd file. In this case, the object does not specify the file location and instead the client uses the file location given by the Device object to find the definition for the object's profile as well as the definition for other profiles used by other objects in the device. Also in this scenario, the vendor has decided to host the xdd file on their web server rather than in the device itself.

BACnet Proprietary Object:
Object_Identifier= (901,1)
Profile_Name="555-ControlRodsObject"

BACnet Device Object:
Profile_Name = "555-BC-Mark-III"
Profile_Location = "http://vendor.example.com/support/defs/BCM3-r1s.xdd"

X.4 Views

A "view" is an abstract concept that is positioned logically above the level of objects and serves to arrange objects and other views into useful hierarchies. A view is implemented as a Structured View object, either physically implemented as an object in a device, or virtually implemented as an <Object> element in an xdd file. Virtually implemented Structured View objects do not appear in an Object_List in any device. Their presence for devices is declared only through xdd files.

Some views are naturally associated with one particular device, either because the majority (or all) of its subordinates are in that device, or simply because the most important ones are. In this case, physically modeling the view would be accomplished with Structured View objects in that device, and virtually modeling the view would be accomplished by using an xdd file for the device that contains virtual instances of Structured View objects.

Some views are not naturally associated with any one particular device, either because there is no device that is primary to the view's content or that primary device is a device that cannot support the Structured View objects, cannot host the xdd files, or may not even have the Profile_Name and Profile_Location properties. In this case, some Structured View object in some other device would contain the physical implementation of the view, or some Device object, in some other device, even in an always-on workstation or web UI server, would reference the xdd file containing the virtual Structured View objects to declare the views that it hosts on behalf of those older devices.

As will be seen in the scenarios below, each of the xdd files can either be a BACnet File on the device that contains the object that is declaring the profile, or located in some other, central, BACnet File repository, or an http-accessible file somewhere.

X.4.1 Factory Fixed-function Scenarios

Scenario 3: A variable frequency drive manufacturer implements the data in their product in some manner and then declares an arrangement of the objects by creating a Structured View object in the device. In this case also, the objects are able to hold their own semantic tagging information using the Tags property.

Note that the Structured View object provides "annotations" for its subordinates that can be used by an HMI in addition to, or instead of, the actual object names.

Structured View Object:
Object_Identifier = (structured-view,1)
Node_Type = equipment
Tags="variable-speed-drive"
Subordinate_List = { (,(binary-value,1)),(,(multistate-value,2)),(,(analog-output,1)), ... }
Subordinate_Annotations = {"Run/Stop Monitor", "Most Recent Fault", "Output Speed", ...}
Subordinate_Node_Types = {point, point, point, ...}

Binary Value Object:
Object_Identifier = (binary-value,1)
Tags = "electric;activity-indicator"

MultiState Value Object:
Object_Identifier = (multistate-value,2)
Tags = "fault-indicator"

Analog Output Object:
Object_Identifier = (analog-output,1)
Tags = "electric;activity-level;command"

Scenario 4: For a product that does not support Structured View objects, the manufacturer declares the same arrangement of the objects by creating a virtual Structured View object in the xdd file referenced by the Device object. Additionally, this product does not support the Tags property and so the xdd file also provides the semantic tags for the subordinates.

```
Device Object:
Profile_Name = "555-VF5000-1.0"
Profile_Location = "bacnet://.this/file,1"

File Object (in same device):
Object_Identifier = (file,1)
Contents of zipped file "ashrae-csml.xml" =
<?xml version="1.0" encoding="UTF-8"?>
<CSML xmlns="http://bacnet.org/csml/1.2">
   ... local definition or "included" definition for  "555-VF5000-1.0"...
   <Object name="drive" virtual="true" >
      <Enumerated name="object-type" value="structured-view"/>
      <ObjectIdentifier name="object-identifier" value="structured-view,1000"/>
      <String name="object-name" value="drive"/>
      <Enumerated name="node-type" value="equipment" />
      <String name="tags" value="variable-speed-drive" />
      <Array name="subordinate-list">
         <Sequence name="1" >
            <ObjectIdentifier name="object-identifier" value="binary-value,1" />
         </Sequence>
         <Sequence name="2">
            <ObjectIdentifier name="object-identifier" value="multistate-value,2" />
         </Sequence>
         <Sequence name="3" >
            <ObjectIdentifier name="object-identifier" value="analog-output,1" />
         </Sequence>
         ... other subordinates ...
      </Array>
      <Array name="subordinate-annotations">
         <String name="1" value="Run/Stop Monitor" />
         <String name="2" value="Most Recent Fault" />
         <String name="3" value="Output Speed" />
         ... other subordinates ...
      </Array>
      <Array name="subordinate-node-types">
         <Enumerated name="1" value="point" />
         <Enumerated name="2" value="point" />
         <Enumerated name="3" value="point" />
         ... other subordinates ...
      </Array>
      <Array name="subordinate-tags">
         <String name="1" value="electric;activity-indicator" />
         <String name="2" value="fault-indicator" />
         <String name="3" value="electric;activity-level;command" />
         ... other subordinates ...
      </Array>
   </Object>
</CSML>
```

X.4.2 Field Applied Scenarios

Scenario 5: A controls contractor programs/configures an applied controller to control an air handler. The programming/configuration tool declares the arrangement of its data using Structured View objects and the Tags property.

The example below shows three typical cases for where the view members' data is located:

(a) the data is in objects internal to this device (most likely);
(b) the data is in objects external to this device;
(c) the data is in another declared Structured View object (external case shown.)

Structured View Object:
Object_Identifier = (structured-view,1)
Subordinate_List = { (,(analog-input,5)),((device,6001),(analog-value,12)),((device,6002),(structured-view,1)) ... }
Subordinate_Annotations = { "Supply Temp", "Supply Static", "Variable Supply Fan" ... }
Tags="equipment;ahu;single-duct"

Analog Input Object:
Object_Identifier = (analog-input,5)
Tags = "point;sensor;temperature;discharge;air"

In Device 6001:
Analog Value Object:
Object_Identifier = (analog-value,12)
Tags = "point;sensor;static-pressure;discharge;air"

In Device 6002:
Structured View Object:
Object_Identifier = (structured-view,1)
Subordinate_List = { ... }
Subordinate_Annotations = { ... }
Tags="section;fan;variable-speed"

Scenario 6: The same information is provided for a device that does not support either the Structure View object or the Tags property. In this case, the Structure View is implemented virtually in the xdd file referenced by the Device object.

BACnet Device Object:
Profile_Name = "555-BCU200-2.0"
Profile_Location = "bacnet://.this/file,22"

BACnet File Object (in same device):
Object_Identifier = (file,22)
Contents of zipped file "ashrae-csml.xml" =

```
<?xml version="1.0" encoding="UTF-8"?>
<CSML xmlns="http://bacnet.org/csml/1.2">
   ... local definition or "included" definition for "555-BCU200-2.0"...
   <Object name="ahu5e" displayName="AHU Fifth Floor East" virtual="true"  >
      <Enumerated name="object-type" value="structured-view"/>
      <ObjectIdentifier name="object-identifier" value="structured-view,1000"/>
      <String name="object-name" value="ahu5e"/>
      <Enumerated name="node-type" value="equipment" />
      <String name="tags" value="ahu;single-duct" />
      <Array name="subordinate-list">
         <Sequence name="1" >
            <ObjectIdentifier name="object-identifier" value="analog-input,5" />
         </Sequence>
         <Sequence name="2">
            <ObjectIdentifier name="device-identifier" value="device,6001" />
            <ObjectIdentifier name="object-identifier" value="analog-value,12" />
         </Sequence>
         <Sequence name="3" >
```

```
            <ObjectIdentifier name="device-identifier" value="device,6002" />
            <ObjectIdentifier name="object-identifier" value="structured-view,1" />
          </Sequence>
        </Array>
        <Array name="subordinate-annotations" >
          <String name="1" value="Supply Temp"/>
          <String name="2" value="Supply Static"/>
          <String name="3" value="Variable Supply Fan" />
        </Array>
        <Array name="subordinate-node-types">
          <Enumerated name="1" value="point" />
          <Enumerated name="2" value="point" />
          <Enumerated name="3" value="section" />
        </Array>
        <Array name="subordinate-tags">
          <String name="1" value="sensor;temperature;discharge;air" />
          <String name="2" value="sensor;staticPressure;discharge;air" />
          <String name="3" value="fan;variable-speed" />
        </Array>
      </Object>
    </CSML>
```

Scenario 7: Instead of hosting the xdd file in a BACnet File object in the programmable controller itself, the installer chooses instead to store the discoverable programming artifact in a central BACnet server with a collection of other such artifacts. This could be because the device does not support BACnet File objects.

Note that in this manner, the view declaration is still "logically hosted" by the programmable controller and therefore "discoverable" at the controller; it's just that the actual file contents are stored remotely.

Also note that the central repository does not list this File object in its own Device object's Profile_Location. That is because it is merely holding the File object for the programmable controller, not "hosting" the view itself.

In controller:
BACnet Device Object:
Profile_Name = "555-BCU200-2.0"
Profile_Location = "bacnet://9000/file,432"

In central repository (BACnet Device 9000):
BACnet File Object:
Object_Identifier = (file,432)
Content={same as scenario 6}

Scenario 8: Same as Scenario 7, but instead of hosting the xdd files in BACnet File objects, the files are hosted on the customer's intranet file server, pointed at by the information in the controller's Device object.

In controller:
BACnet Device Object:
Profile_Name = "555-BCU200-2.0"
Profile_Location = "http://bws.customer.example.com/discovery/xdd/dev-3001-infs.xdd"

In central repository (http file server):
Http File: http://bws.customer.example.com/discovery/xdd/dev-3001-infs.xdd
Content=
{same as scenario 6}

Scenario 9: The programmable controller is an older device or one that does not have a usable Profile_Location or Deployed_Profile_Location property. This situation can happen when these properties are either not present, not writable, not writable with sufficient length, or cannot be otherwise configured to point to the appropriate location. In this case, the programming artifacts must be stored on another device or central repository. When the client finds this older programmable controller (e.g., via a Who-Is scan), it has no direct way to "discover" views implemented on it. However, after either scanning the entire network, or being given a hint as to a central repository's location, the client will eventually find the programming artifact files and will thus indirectly "discover" the views declared for the programmable controller.

When the xdd files are stored remotely this way, there is no implicit link to the programmable device for which they are intended to be associated with. To create such an association, an xdd file can contain an explicit reference to a device by including an object with an "object-identifier" property that references the device.

Since this central repository is hosting xdd files for multiple controllers, but can only have a single xdd file itself, the implementer of the repository has a choice. If the number of hosted controllers is small, it can simply merge their xdd information into a single file. However, it may be desirable for this repository to maintain its single xdd file with nothing but links to other individual xdd files, possibly one for each controller. This allows the size of this single "directory" file to be kept reasonable and allows the client to access the individual xdd files one at a time. This is the scenario shown in the example below.

In older programmable controller:
BACnet Device Object:
...nothing...

In central repository (or any other "hosting" device):
BACnet Device Object:
Profile_Name = "555-OWS-5.0"
Profile_Location = "http://bws.customer.example.com/discovery/xdd/all.xdd"

In central repository (http file server):
Http File: http://bws.customer.example.com/discovery/xdd/all.xdd
Contents of zipped file "ashrae-csml.xml" =

```
<?xml version="1.0" encoding="UTF-8"?>
<CSML xmlns="http://bacnet.org/csml/1.2">
   ... local definition for "555-OWS-5.0"...
</CSML>
```

Contents of zipped file "ashrae-links.txt" =

```
Link: <east/6000.xdd>; rel="related"
Link: <east/6001.xdd>; rel="related"
Link: <west/6002.xdd>; rel="related"
```

Note that the locations of the above linked xdd files are relative to the location of the "all.xdd" file, i.e. the first linked file is at http://bws.customer.example.com/discovery/xdd/east/6000.xdd.

In central repository (http file server):
Http File: http://bws.customer.example.com/discovery/xdd/east/6000.xdd
Content=

```
{ similar to scenario 6, plus:
<Object><ObjectIdentifier name="object-identifier" value="device,6000"/></Object> }
```

X.4.3 Additional Deployment Scenarios

A customer or a controls contractor might wish to provide additional semantic or arrangement information beyond that provided by a factory integrated controller or a previously applied controller. In order to not affect the existing definitions and views defined by the xdd files referred to by the Profile_Location properties in those devices, the additional information can be provided in separate xdd files and referred to by the Deployed_Profile_Location property.

Scenario 10: A collection of controllers is involved in building up a subsystem, like a chiller plant or a micro-grid. The customer creates an xdd file containing all the virtual Structured View objects that convey the arrangement of the controllers' existing objects and views into new higher level views. The customer then chooses a single controller to point to this information. To avoid having to merge this information into the existing controller's profile information, the customer uses the Deployed_Profile_Location property and leaves the information pointed to by the controller's Profile_Location unchanged.

BACnet Device Object:
Profile_Name = "555-BCU200-2.0" (unchanged)
Profile_Location = "bacnet://.this/file,22" (unchanged)
Deployed_Profile_Location = "http://bws.customer.example.com/deployed/chillerplant.xdd"

BACnet File Object (in same device):
Object_Identifier = (file,22)
Content=
{ describes only the profiles and views programmed into the single device }

Http File: http://bws.customer.example.com/deployed/chillerplant.xdd
Contents of zipped file "ashrae-csml.xml" =

```
<?xml version="1.0" encoding="UTF-8"?>
<CSML xmlns="http://bacnet.org/csml/1.2">
  <Object name="chwplant" displayName="Chiller Plant" >
    ... subordinates arranging multiple controller's objects and views into a chiller plant ...
  </Object>
  ... more views ...
</CSML>
```

X.5 PICS Declarations

To enable devices to declare their capabilities in a machine readable format, a reserved section of the device's xdd file is defined to hold PICS information in XML. The PICS information shall be placed in a <Composition> element directly under the <CSML> element with a 'name' attribute equal to ".pics." The CSML type of this element shall be one of the standard PICS types defined by ASHRAE that is for a protocol revision that is greater than or equal to the Protocol_Revision of the device.

For example, the xdd file referenced from the Device object's Profile_Location might contain an "ashrae-csml.xml" file that might contain:

```
<?xml version="1.0" encoding="UTF-8"?>
<CSML xmlns="http://bacnet.org/csml/1.2">
  ...
  <Composition name=".pics" type="0-PICS" > <!-- example type only, refer to separate definition -->
    <!-- example members only, actual type is defined separately from this standard -->
    <String name="vendor-name" value="Controls-R-Us"/>
    <String name="product-name" value="Building Controller Mark III"/>
    <String name="product-description" value="A really great thing"/>
    <Unsigned name="vendor-identifier" value="555"/>
    ... more ...
  </Composition>
  ...
</CSML>
```

[Change all **Tables 12-?"**Properties of ... Object Type" for all object types in Clause 12]

Table 12-?. Properties of ... Object Type

Property Identifier	Property Datatype	Conformance Code
...	...	...
Profile_Location	*CharacterString*	*O*

[Insert new **Clause 12.?.X1** to all object types in Clause 12]

12.?.X1 Profile_Location

This property, of type CharacterString, is the URI of the location of an xdd file (See Clause X.2) containing the definition of the CSML type specified by the Profile_Name property and possible other information (See Annex X). The URI is restricted to using only the "http", "https", and "bacnet" URI schemes. See Clause Q.8 for the definition of the "bacnet" URI scheme.

If a Profile_Location value is not provided for a particular object, then the client shall use the Profile_Location of the Device object, if provided, to find the definition of the Profile_Name.

[Change **Table 12-13**, p. 200]

Table 12-13. Properties of the Device Object Type

Property Identifier	Property Datatype	Conformance Code
...	...	...
Deployed_Profile_Location	*CharacterString*	*O*[n]

[n] *this property is required to be writable if present*

[Insert new **Clause 12.11.X**, p. 207]

12.11.X Deployed_Profile_Location

This property, of type CharacterString, is the URI of the location of an xdd file (See Clause X.2). The URI is restricted to using only the "http", "https", and "bacnet" URI schemes. See Clause Q.8 for the definition of the "bacnet" URI scheme. The referenced xdd file contains additional information about the deployed device. It is intended to be used as a supplement to the information referenced by the Profile_Location property. If present, this property shall be writable and shall, at a minimum, support storage of strings with an encoded length up to 255 octets.

[Change all **Clauses 12.?.? Profile_Name**]

12.?.? Profile_Name

This property, of type CharacterString, is the name of an object profile to which this object conforms. To ensure uniqueness, a profile name shall begin with a vendor identifier code (see Clause 23) in base-10 integer format, followed by a dash. All subsequent characters are administered by the organization registered with that vendor identifier code. The vendor identifier code that prefixes the profile name shall indicate the organization that publishes and maintains the profile ~~document named by the remainder of the profile name~~. This vendor identifier need not have any relationship to the vendor identifier of the device within which the object resides.

A profile defines a set of additional properties, behavior, and/or requirements for this object beyond those specified here. ~~This standard defines only the format of the names of profiles. The definition of the profiles themselves is outside the scope of this standard.~~ *If the Profile_Location property of this object or the Device object is present and nonempty, then the value of this property shall be the name of a CSML type defined in an xdd file referred to by the Profile_Location property.*

135-2012*ba*-2. Add Semantic Tags to All Objects.

[This section makes use of additions and changes that are defined in Addendum *am*. Therefore, it is imperative that Addendum 135-2012*am* sections 2, 3, and 6 be applied to the standard before applying this addendum.]

Rationale

Determining the "meaning" of a BACnet object can sometimes be inferred by humans based on its name or description, but this process is imprecise. Interoperability will be greatly enhanced by the presence of standardized semantic information on an object, which can allow consumers (humans and machines) to make presentation, reporting, grouping, and operational decisions based on the meaning of the object.

[Change all **Table 12-? "**Properties of ... Object Type" for all object types in Clause 12]

Table 12-?. Properties of ... Object Type

Property Identifier	Property Datatype	Conformance Code
...	...	...
Tags	*BACnetARRAY[N] of BACnetNameValue*	*O*

[Insert **Clause 12.?.Y1** to all object types in Clause 12]

12.?.Y1 Tags

This property, of type BACnetARRAY of BACnetNameValue, is a collection of tags for the object. See Clause Y.1.4 for restrictions on the string values used for the names of these tag and for a description of tagging and the mechanism by which tags are defined.

Each entry in the array is a BACnetNameValue construct which consists of the tag name and an optional value. If the tag is defined to be a "semantic tag" then it has no value, and the "value" field of the BACnetNameValue shall be absent.

While some tags may be known in advance when a device is manufactured, it is recommended that implementations consider that this kind of information might not be known until a device is deployed and to provide a means of configuration or writability of this property.

[Add new production to **Clause 21**, p. 687]

```
BACnetNameValue ::= SEQUENCE {
        name            [0] CharacterString,
        value           ABSTRACT-SYNTAX.&Type OPTIONAL
                        -- value is limited to primitive datatypes and BACnetDateTime
        }
```

135-2012*ba*-3. Extend Structured View Object to Contain Semantic Information.

[This section makes use of additions and changes that are defined in Addendum *am*. Therefore, it is imperative that Addendum 135-2012*am* sections 2, 3, and 6 be applied to the standard before applying this addendum.]

Rationale

The addition of the Tags property in Section 2 of this addendum allows objects to express a semantic meaning. However, not all objects will have this property or may have restrictions on its configuration; therefore, this section adds the ability for Structured View objects to be able to provide additional semantic information for its subordinates.

[Change **Table 12-34**, p. 314]

Table 12-34. Properties of the Structured View Object Type

Property Identifier	Property Datatype	Conformance Code
...	...	...
Subordinate_Tags	*BACnetARRAY[N] of BACnetNameValueCollection*	*O*
Subordinate_Node_Types	*BACnetARRAY[N] of BACnetNodeType*	*O*
Subordinate_Relationships	*BACnetARRAY[N] of BACnetRelationship*	*O*
Default_Subordinate_Relationship	*BACnetRelationship*	*O*
Represents	*BACnetDeviceObjectReference*	*O*
...	...	...

[Add new **Clauses to 12.29**, p. 315]

12.29.X1 Subordinate_Tags

This property, of type BACnetARRAY of BACnetNameValueCollection, provides individual collections of tags for each of the subordinates. See Clause Y.1.4 for restrictions on the string values used for the names of these tags and a general description of tagging and the mechanism by which tags are defined.

If the size of this array is changed, the size of the Subordinate_List, Subordinate_Annotations, Subordinate_Node_Types, and Subordinate_Relationships arrays, if present, shall also be changed to the same size. Uninitialized Subordinate_Tags array elements shall be empty collections.

12.29.X2 Subordinate_Node_Types

This property, of type BACnetARRAY of BACnetNodeType, shall be used to provide node type (Clause 12.29.5) information for each member of the Subordinate_List. If the subordinate object has its own Node_Type, it is intended that the local value in this property logically override the subordinate's Node_Type unless the local value is "UNKNOWN", in which case, the subordinate's Node_Type is used.

If the size of this array is changed, the size of the Subordinate_List, Subordinate_Annotations, Subordinate_Tags, and Subordinate_Relationships arrays, if present, shall also be changed to the same size. Uninitialized Subordinate_Node_Types array elements shall have the value UNKNOWN.

12.29.X3 Subordinate_Relationships

This property, of type BACnetARRAY of BACnetRelationship, shall be used to describe the relationship to each member of the Subordinate_List. If this property is absent, then relationship to each of the subordinates is equal to the value of the Default_Subordinate_Relationship property, if present, else equal to UNKNOWN.

If the size of this array is changed, the size of the Subordinate_List, Subordinate_Annotations, Subordinate_Tags, and Subordinate_Node_Types arrays, if present, shall also be changed to the same size. Uninitialized Subordinate_Relationships array elements shall be equal to DEFAULT.

12.29.X4 Default_Subordinate_Relationship

This property, of type BACnetRelationship, shall be used to describe the default relationship to each member of the Subordinate_List, unless overridden by a member of the Subordinate_Relationships property. If this property is absent and the Subordinate_Relationships is also absent, the relationship to the subordinates is equal to UNKNOWN.

12.29.X5 Represents

This property, of type BACnetDeviceObjectReference, can be used to indicate the entity for which this Structured View is modeling the "subordinates".

In some cases, the Structure View object will abstractly represent this entity by itself, and this property will either be absent, unconfigured, or point to itself. For example, this could be a Structured View object that abstractly represents a simple rooftop unit, where the subordinates are its points. This Structured View therefore logically "is" the rooftop unit and thus is the parent of the subordinates.

In other cases, there may be a separate concrete object that represents the entity. This Structured View therefore logically provides the subordinates for that other object. For example, the other object could be another Structured View, a Program object, a Load Control object, or any other object that best represents the entity. In this case, the Represents property can be configured to point to that object and the subordinates configured in this Structured View are then logically subordinates of that object.

In this manner, multiple Structured View objects can be used to provide multiple sets of subordinates for a single entity, thereby providing deployment flexibility. For example, if an entity has relationship X with one set of subordinates and has relationship Y with another set of subordinates, deployments can, among other possibilities:

(a) use one Structured View object with all the subordinates combined into a single array and use the Subordinate_Relationships property to select between X and Y for each of the subordinates, or
(b) use two Structured View objects, one for the subordinates with relationship X and another for those with relationship Y. Since the relationships are segregated, the Default_Subordinate_Relationship property can be used in each of the two views to select the relationship for all of that view's subordinates, and the Represents property of the two views can be used to tie them to a common entity.

[Change **Clause 12.29.7**, p. 315]

12.29.7 Subordinate_List

...
If the size of the Subordinate_List array is changed, the size of the Subordinate_Annotations*, Subordinate_Tags, Subordinate_Relationships, and Subordinate_Node_Types* ~~array~~ *arrays*, if present, shall also be changed to the same size. Uninitialized Subordinate_List array elements shall be given the instance number 4194303.
...

[Change **Clause 12.29.8**, p. 315]

12.29.8 Subordinate_Annotations

...
If the size of this array is changed, the size of the Subordinate_List*, Subordinate_Tags, Subordinate_Relationships, and Subordinate_Node_Types* ~~array~~ *arrays* shall also be changed to the same size.

[Add new production to **Clause 21**, p. 687]

BACnetNameValueCollection ::= SEQUENCE {
    members   [0]   SEQUENCE OF BACnetNameValue
    }

[Change production in **Clause 21**, p. 694]

BACnetPropertyIdentifier ::= ENUMERATED { -- see below for numerical order

```
    ...
    default-step-increment                  (376),
    default-subordinate-relationship        (490),
    deployed-profile-location               (484),
    derivative-constant                     (26),
    ...
    process-identifier-filter               (361),
    profile-location                        (485),
    profile-name                            (168),
    ...
    relinquish-default                      (104),
    represents                              (491),
    requested-shed-level                    (218),
    ...
    subordinate-list                        (211),
    subordinate-node-types                  (487),
    subordinate-relationship                (489),
    subordinate-tags                        (488),
    subscribed-recipients                   (362),
    ...
    system-status                           (112),
    tags                                    (486),
    threat-authority                        (306),
    ...
    -- see transition                       (385),
    ...
    -- see deployed-profile-location        (484),
    -- see profile-location                 (485),
    -- see tags                             (486),
    -- see subordinate-node-types           (487),
    -- see subordinate-tags                 (488),
    -- see subordinate-relationship         (489),
    -- see default-subordinate-relationship (490),
    -- see represents                       (491),
    ...
    }
```

135-2012*ba*-4. Change Clause 21 identifiers to use a consistent format.

[Addendum 135-2012*am* shall be applied to the standard before applying the changes in this addendum. Therefore, this section has been removed from this addendum because it has been superceded by changes made in 135-2012*am*-6.]

135-2012*ba*-5. Add Data Revisioning Capabilities to CSML.

[Addendum 135-2012*am* shall be applied to the standard before applying the changes in this addendum. Therefore, this section has been removed from this addendum because it has been superceded by changes made in 135-2012*am*-2.]

[Add a new entry to **History of Revisions**, p. 1027]

(This History of Revisions is not part of this standard. It is merely informative and does not contain requirements necessary for conformance to the standard.)

HISTORY OF REVISIONS

…	…	**…**
1	19	**Addendum ba to ANSI/ASHRAE 135-2012** Approved by the ASHRAE Standards Committee on April 29, 2016, and by the American National Standards Institute on April 29, 2016. 1. Add CSML Descriptions of BACnet Devices 2. Add Semantic Tags to All Objects 3. Extend Structured View Object to Contain Semantic Information 4. Change Clause 21 identifiers to use a consistent format 5. Add Data Revisioning Capabilities to CSML

POLICY STATEMENT DEFINING ASHRAE'S CONCERN FOR THE ENVIRONMENTAL IMPACT OF ITS ACTIVITIES

ASHRAE is concerned with the impact of its members' activities on both the indoor and outdoor environment. ASHRAE's members will strive to minimize any possible deleterious effect on the indoor and outdoor environment of the systems and components in their responsibility while maximizing the beneficial effects these systems provide, consistent with accepted Standards and the practical state of the art.

ASHRAE's short-range goal is to ensure that the systems and components within its scope do not impact the indoor and outdoor environment to a greater extent than specified by the Standards and Guidelines as established by itself and other responsible bodies.

As an ongoing goal, ASHRAE will, through its Standards Committee and extensive Technical Committee structure, continue to generate up-to-date Standards and Guidelines where appropriate and adopt, recommend, and promote those new and revised Standards developed by other responsible organizations.

Through its *Handbook*, appropriate chapters will contain up-to-date Standards and design considerations as the material is systematically revised.

ASHRAE will take the lead with respect to dissemination of environmental information of its primary interest and will seek out and disseminate information from other responsible organizations that is pertinent, as guides to updating Standards and Guidelines.

The effects of the design and selection of equipment and systems will be considered within the scope of the system's intended use and expected misuse. The disposal of hazardous materials, if any, will also be considered.

ASHRAE's primary concern for environmental impact will be at the site where equipment within ASHRAE's scope operates. However, energy source selection and the possible environmental impact due to the energy source and energy transportation will be considered where possible. Recommendations concerning energy source selection should be made by its members.

ASHRAE · 1791 Tullie Circle NE · Atlanta, GA 30329 · www.ashrae.org

About ASHRAE

ASHRAE, founded in 1894, is a global society advancing human well-being through sustainable technology for the built environment. The Society and its members focus on building systems, energy efficiency, indoor air quality, refrigeration, and sustainability. Through research, Standards writing, publishing, certification and continuing education, ASHRAE shapes tomorrow's built environment today.

For more information or to become a member of ASHRAE, visit www.ashrae.org.

To stay current with this and other ASHRAE Standards and Guidelines, visit www.ashrae.org/standards.

Visit the ASHRAE Bookstore

ASHRAE offers its Standards and Guidelines in print, as immediately downloadable PDFs, on CD-ROM, and via ASHRAE Digital Collections, which provides online access with automatic updates as well as historical versions of publications. Selected Standards and Guidelines are also offered in redline versions that indicate the changes made between the active Standard or Guideline and its previous version. For more information, visit the Standards and Guidelines section of the ASHRAE Bookstore at www.ashrae.org/bookstore.

IMPORTANT NOTICES ABOUT THIS STANDARD

To ensure that you have all of the approved addenda, errata, and interpretations for this Standard, visit www.ashrae.org/standards to download them free of charge.

Addenda, errata, and interpretations for ASHRAE Standards and Guidelines are no longer distributed with copies of the Standards and Guidelines. ASHRAE provides these addenda, errata, and interpretations only in electronic form to promote more sustainable use of resources.

4/16