# Annex A - HTTP Bulk Data Collection

*Note - This Annex is a translation from the HTTP Bulk Data Collection mechanism specified in Annex A of* [*Broadband Forum TR-157*](https://www.broadband-forum.org/technical/download/TR-157.pdf)*, which was carried over into Amendment 6 of* [*TR-069*](https://www.broadband-forum.org/technical/download/TR-069.pdf)*. The text here has been altered to fit with USP concepts.*

This section discusses the Theory of Operation for the collection and transfer of bulk data using USP, HTTP and the BulkData object defined in [Device:2](https://github.com/BroadbandForum/usp/tree/master/data-model), to a Bulk Data Collector utilizing:

* HTTP/HTTPS for the transfer of collected data
* CSV and JSON for the encoding of collected data to be transferred

The Agent configuration that enables the collection of bulk data using HTTP is defined using the BulkData component objects explained here. During this explanation, there will be references to data model objects specific to [Device:2](https://github.com/BroadbandForum/usp/tree/master/data-model); that specification should be used for reference.

## Enabling HTTP/HTTPS Bulk Data Communication

HTTP/HTTPS communication between the Agent and Bulk Data Collector is enabled by configuring the BulkData.Profile object for the HTTP/HTTPS transport protocol adding and configuring a new BulkData.Profile object instance using the [Add](/specification/messages/add/) message. For example:

.BulkData.Profile.1  
.BulkData.Profile.1.Enable=true  
.BulkData.Profile.1.Protocol = "HTTP"  
.BulkData.Profile.1.ReportingInterval = 300  
.BulkData.Profile.1.TimeReference = "0001-01-01T00:00:00Z"  
.BulkData.Profile.1.HTTP.URL = "https://bdc.acme.com/somedirectory"  
.BulkData.Profile.1.HTTP.Username = "username"  
.BulkData.Profile.1.HTTP.Password = "password"  
.BulkData.Profile.1.HTTP.Method = "POST"  
.BulkData.Profile.1.HTTP.UseDateHeader = true

The configuration above defines a profile that transfers data from the Agent to the Bulk Data Collector using secured HTTP. In addition the Agent will provide authentication credentials (username, password) to the Bulk Data Collector, if requested by the Bulk Data Collector. Finally, the Agent establishes a communication session with the Bulk Data Collector every 300 seconds in order to transfer the data defined by the .BulkData.Report. object instance.

Once the communication session is established between the Agent and Bulk Data Collector the data is transferred from the Agent using the POST HTTP method with a HTTP Date header and no compression.

**R-BULK.0** - In many scenarios Agents will utilize "chunked" transfer encoding. As such, the Bulk Data Collector MUST support the HTTP transfer-coding value of "chunked".

### Use of the URI Query Parameters

The HTTP Bulk Data transfer mechanism allows parameters to be used as HTTP URI query parameters. This is useful when Bulk Data Collector utilizes the specific parameters that the Agent reports for processing (e.g., logging, locating directories) without the need for the Bulk Data Collector to parse the data being transferred.

**R-BULK.1** - The Agent MUST transmit the device's Manufacturer OUI, Product Class and Serial Number as part of the URI query parameters. The data model parameters are encoded as:

.DeviceInfo.ManufacturerOUI -> oui  
.DeviceInfo.ProductClass -> pc  
.DeviceInfo.SerialNumber -> sn

As such, the values of the device’s OUI, Serial Number and Product Class are formatted in the HTTP request URI as follows:

POST https://<bulk data collector url>?oui=00256D&pc=Z&sn=Y

Configuring the URI query parameters for other parameters requires that instances of a .BulkData.Profile.{i}.HTTP.RequestURIParameter object instance be created and configured with the requested parameters. The additional parameters are appended to the required URI query parameters.

Using the example to add the device's current local time to the required URI parameters, the HTTP request URI would be as follows:

POST https://<bulk data collector url>?oui=00256D&pc=Z&sn=Y&ct=2015-11-01T11:12:13Z

By setting the following parameters using the Add message as follows:

.BulkData.Profile.1.HTTP.RequestURIParameter 1.Name ="ct"  
.BulkData.Profile.1.HTTP.RequestURIParameter.1.Reference ="Device.Time.CurrentLocalTime"

### Use of HTTP Status Codes

The Bulk Data Collector uses standard HTTP status codes, defined in the HTTP specification, to inform the Agent whether a bulk data transfer was successful. The HTTP status code is set in the response header by the Bulk Data Collector. For example, "200 OK" status code indicates an upload was processed successfully, "202 Accepted" status code indicates that the request has been accepted for processing, but the processing has not been completed, "401 Unauthorized" status code indicates user authentication failed and a "500 Internal Server Error" status code indicates there is an unexpected system error.

#### HTTP Retry Mechanism

**R-BULK.2** - When the Agent receives an unsuccessful HTTP status code and the HTTP retry behavior is enabled, the Agent MUST try to redeliver the data. The retry mechanism employed for the transfer of bulk data using HTTP uses the same algorithm as is used for [USP Notify retries](/messages/#notifications).

The retry interval range is controlled by two Parameters, the minimum wait interval and the interval multiplier, each of which corresponds to a data model Parameter, and which are described in the table below. The factory default values of these Parameters MUST be the default values listed in the Default column. They MAY be changed by a Controller with the appropriate permissions at any time.

|  |  |  |  |
| --- | --- | --- | --- |
| Descriptive Name | Symbol | Default | Data Model Parameter Name |
| Minimum wait interval | m | 5 seconds | Device.BulkData.Profile.{i}.HTTP.RetryMinimumWaitInterval |
| Interval multiplier | k | 2000 | Device.BulkData.Profile.{i}.HTTP.RetryIntervalMultiplier |

|  |  |  |
| --- | --- | --- |
| Retry Count | Default Wait Interval Range (min-max seconds) | Actual Wait Interval Range (min-max seconds) |
| #1 | 5-10 | m - m.(k/1000) |
| #2 | 10-20 | m.(k/1000) - m.(k/1000)2 |
| #3 | 20-40 | m.(k/1000)2 - m.(k/1000)3 |
| #4 | 40-80 | m.(k/1000)3 - m.(k/1000)4 |
| #5 | 80-160 | m.(k/1000)4 - m.(k/1000)5 |
| #6 | 160-320 | m.(k/1000)5 - m.(k/1000)6 |
| #7 | 320-640 | m.(k/1000)6 - m.(k/1000)7 |
| #8 | 640-1280 | m.(k/1000)7 - m.(k/1000)8 |
| #9 | 1280-2560 | m.(k/1000)8 - m.(k/1000)9 |
| #10 and subsequent | 2560-5120 | m.(k/1000)9 - m.(k/1000)10 |

**R-BULK.3** - Beginning with the tenth retry attempt, the Agent MUST choose from the fixed maximum range. The Agent will continue to retry a failed bulk data transfer until it is successfully delivered or until the next reporting interval for the data transfer becomes effective.

**R-BULK.4** - Once a bulk data transfer is successfully delivered, the Agent MUST reset the retry count to zero for the next reporting interval.

**R-BULK.5** - If a reboot of the Agent occurs, the Agent MUST reset the retry count to zero for the next bulk data transfer.

### Use of TLS and TCP

The use of TLS to transport the HTTP Bulk Data is RECOMMENDED, although the protocol MAY be used directly over a TCP connection instead. If TLS is not used, some aspects of security are sacrificed. Specifically, TLS provides confidentiality and data integrity, and allows certificate-based authentication in lieu of shared secret-based authentication.

**R-BULK.6** - Certain restrictions on the use of TLS and TCP are defined as follows:

* The Agent MUST support TLS version 1.2 or later.
* If the Collection Server URL has been specified as an HTTPS URL, the Agent MUST establish secure connections to the Collection Server, and MUST start the TLS session negotiation with TLS 1.2 or later.

*Note - If the Collection Server does not support the version with which the Agent establishes the connection, it might be necessary to negotiate an earlier TLS 1.x version, or even SSL 3.0. This implies that the Agent has to support the mandatory cipher suites for all supported TLS or SSL versions.*

*Note - TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA is the only mandatory TLS 1.2 cipher suite.*

* The Agent SHOULD use the [RFC 6066](https://tools.ietf.org/html/rfc6066) Server Name TLS extension to send the host portion of the Collection Server URL as the server name during the TLS handshake.
* If TLS 1.2 (or a later version) is used, the Agent MUST authenticate the Collection Server using the certificate provided by the Collection Server. Authentication of the Collection Server requires that the Agent MUST validate the certificate against a root certificate. To validate against a root certificate, the Agent MUST contain one or more trusted root certificates that are either pre-loaded in the Agent or provided to the Agent by a secure means outside the scope of this specification. If as a result of an HTTP redirect, the Agent is attempting to access a Collection Server at a URL different from its pre-configured Collection Server URL, the Agent MUST validate the Collection Server certificate using the redirected Collection Server URL rather than the pre-configured Collection Server URL.
* If the host portion of the Collection Server URL is a DNS name, this MUST be done according to the principles of [RFC 6125](https://tools.ietf.org/html/rfc6125), using the host portion of the Collection Server URL as the reference identifier.
* If the host portion of the Collection Server URL is an IP address, this MUST be done by comparing the IP address against any presented identifiers that are IP addresses.

*Note - the terms "reference identifier" and "presented identifier" are defined in* [*RFC 6125*](https://tools.ietf.org/html/rfc6125)*.* *Note - wildcard certificates are permitted as described in* [*RFC 6125*](https://tools.ietf.org/html/rfc6125)

* A Agent capable of obtaining absolute time SHOULD wait until it has accurate absolute time before contacting the Collection Server. If a Agent for any reason is unable to obtain absolute time, it can contact the Collection Server without waiting for accurate absolute time. If a Agent chooses to contact the Collection Server before it has accurate absolute time (or if it does not support absolute time), it MUST ignore those components of the Collection Server certificate that involve absolute time, e.g. not-valid-before and not-valid-after certificate restrictions.
* Support for Agent authentication using client-side certificates is NOT RECOMMENDED. Instead, the Collection Server SHOULD authenticate the Agent using HTTP basic or digest authentication to establish the identity of a specific Agent.

## Encoding of Bulk Data

Bulk Data that is transferred to the Bulk Data Collector from the Agent using HTTP/HTTPS is encoded using a specified encoding type. For HTTP/HTTPS the supported encoding types are CSV and JSON. The encoding type is sent a media type with the report format used for the encoding. For CSV the media type is text/csv as specified in [RFC 4180](https://tools.ietf.org/html/rfc4180) and for JSON the media type is application/json as specified in [RFC 7159](https://tools.ietf.org/html/rfc7159). For example, a CSV encoded report using charset=UTF-8 would have the following Content-Type header:

Content-Type: text/csv; charset=UTF-8

**R-BULK.7** - The "media-type" field and "charset" parameters MUST be present in the Content-Type header.

In addition the report format that was used for encoding the report is included as a HTTP custom header with the following format:

BBF-Report-Format: <ReportFormat>

The field is represented as a token.

For example a CSV encoded report using a ReportFormat for ParameterPerRow would have the following BBF-Report-Format header:

BBF-Report-Format: "ParameterPerRow"

**R-BULK.8** - The BBF-Report-Format custom header MUST be present when transferring data to the Bulk Data Collector from the Agent using HTTP/HTTPS.

### Using Wildcards to Reference Object Instances in the Report

When the Agent supports the use of the Wildcard value "\*" in place of instance identifiers for the Reference parameter, then all object instances of the referenced parameter are encoded. For example to encode the "BroadPktSent" parameter for all object instances of the MoCA Interface object the following will be configured:

.BulkData.Profile.1.Parameter.1.Name = ""  
 .BulkData.Profile.1.Parameter.1.Reference = "Device.MoCA.Interface.\*.Stats.BroadPktSent"

### Using Alternative Names in the Report

Alternative names can be defined for the parameter name in order to shorten the name of the parameter. For example instead of encoding the full parameter name "Device.MoCA.Interface.1.Stats.BroadPktSent" could be encoded with a shorter name "BroadPktSent". This allows the encoded data to be represented using the shorter name. This would be configured as:

.BulkData.Profile.1.Parameter.1.Name = "BroadPktSent"  
.BulkData.Profile.1.Parameter.1.Reference = "Device.MoCA.Interface.1.Stats.BroadPktSent"

In the scenario where there are multiple instances of a parameter (e.g., "Device.MoCA.Interface.1.Stats.BroadPktSent", "Device.MoCA.Interface.2.Stats.BroadPktSent") in a Report, the content of the Name parameter SHOULD be unique (e.g., BroadPktSent1, BroadPktSent2).

#### Using Object Instance Wildcards and Parameter Partial Paths with Alternative Names

Wildcards for Object Instances can be used in conjunction with the use of alternative names by reflecting object hierarchy of the value of the Reference parameter in the value of the Name parameter.

**R-BULK.9** - When the value of the Reference parameter uses a wildcard for an instance identifier, the value of the Name parameter (as used in a report) MUST reflect the wild-carded instance identifiers of the parameters being reported on. Specifically, the value of the Name parameter MUST be appended with a period (.) and then the instance identifier. If the value of the Reference parameter uses multiple wildcard then each wild-carded instance identifier MUST be appended in order from left to right.

For example, for a device to report the Bytes Sent for the Associated Devices of the device's WiFi Access Points the following would be configured:

.BulkData.Profile.1.Parameter.1.Name = "WiFi\_AP\_Assoc\_BSent"  
.BulkData.Profile.1.Parameter.1.Reference = "Device.WiFi.AccessPoint.\*.AssociatedDevice.\*.Stats.BytesSent"

Using this configuration a device that has 2 WiFi Access Points (with instance identifiers 1 and 3) each with 2 Associated Devices (with instance identifiers 10 and 11), would contain a Report with following parameter names:

WiFi\_AP\_Assoc\_BSent.1.10  
WiFi\_AP\_Assoc\_BSent.1.11  
WiFi\_AP\_Assoc\_BSent.3.10  
WiFi\_AP\_Assoc\_BSent.3.11

Object or Object Instance paths can also be used to report all parameters of the associated Object.

**R-BULK.10** - When the value of the Reference parameter is an Object Path, the value of the Name parameter (as used in a report) MUST reflect the remainder of the parameter path. Specifically, the value of Name parameter MUST be appended with a "." and then the remainder of the parameter path.

For example, for a device to report the statistics of a WiFi associated device object instance the following would be configured:

.BulkData.Profile.1.Parameter.1.Name = " WiFi\_AP1\_Assoc10"  
.BulkData.Profile.1.Parameter.1.Reference = "Device.WiFi.AccessPoint.1.AssociatedDevice.10.Stats."

Using the configuration the device's report would contain the following parameter names:

WiFi\_AP1\_Assoc10.BytesSent  
WiFi\_AP1\_Assoc10.BytesReceived  
WiFi\_AP1\_Assoc10.PacketsSent  
WiFi\_AP1\_Assoc10.PacketsReceived  
WiFi\_AP1\_Assoc10.ErrorsSent  
WiFi\_AP1\_Assoc10.RetransCount  
WiFi\_AP1\_Assoc10.FailedRetransCount  
WiFi\_AP1\_Assoc10.RetryCount  
WiFi\_AP1\_Assoc10.MultipleRetryCount

It is also possible for the value of the Reference parameter to use both wildcards for instance identifiers and be a partial path. For example, for device to report the statistics for the device's WiFi associated device, the following would be configured:

.BulkData.Profile.1.Parameter.1.Name = "WiFi\_AP\_Assoc"  
.BulkData.Profile.1.Parameter.1.Reference = "Device.WiFi.AccessPoint.\*.AssociatedDevice.\*.Stats."

Using this configuration a device that has 1 WiFi Access Point (with instance identifier 10) with 2 Associated Devices (with instance identifiers 10 and 11), would contain a Report with following parameter names:

WiFi\_AP\_Assoc.1.10.BytesSent  
WiFi\_AP\_Assoc.1.10.BytesReceived  
WiFi\_AP\_Assoc.1.10.PacketsSent  
WiFi\_AP\_Assoc.1.10.PacketsReceived  
WiFi\_AP\_Assoc.1.10.ErrorsSent  
WiFi\_AP\_Assoc.1.10.RetransCount  
WiFi\_AP\_Assoc.1.10.FailedRetransCount  
WiFi\_AP\_Assoc.1.10.RetryCount  
WiFi\_AP\_Assoc.1.10.MultipleRetryCount  
WiFi\_AP\_Assoc.1.11.BytesSent  
WiFi\_AP\_Assoc.1.11.BytesReceived  
WiFi\_AP\_Assoc.1.11.PacketsSent  
WiFi\_AP\_Assoc.1.11.PacketsReceived  
WiFi\_AP\_Assoc.1.11.ErrorsSent  
WiFi\_AP\_Assoc.1.11.RetransCount  
WiFi\_AP\_Assoc.1.11.FailedRetransCount  
WiFi\_AP\_Assoc.1.11.RetryCount  
WiFi\_AP\_Assoc.1.11.MultipleRetryCount

### Processing of Content for Failed Report Transmissions

When the content (report) cannot be successfully transmitted, including retries, to the data collector, the NumberOfRetainedFailedReports parameter of the BulkData.Profile object instance defines how the content should be disposed based on the following rules:

* When the value of the NumberOfRetainedFailedReports parameter is greater than 0, then the report for the current reporting interval is appended to the list of failed reports. How the content is appended is dependent on the type of encoding (e.g., CSV, JSON) and is described further in corresponding encoding section.
* If the value of the NumberOfRetainedFailedReports parameter is -1, then the Agent will retain as many failed reports as possible.
* If the value of the NumberOfRetainedFailedReports parameter is 0, then failed reports are not to be retained for transmission in the next reporting interval.
* If the Agent cannot retain the number of failed reports from previous reporting intervals while transmitting the report of the current reporting interval, then the oldest failed reports are deleted until the Agent is able to transmit the report from the current reporting interval.
* If the value BulkData.Profile object instance’s EncodingType parameter is modified any outstanding failed reports are deleted.

### Encoding of CSV Bulk Data

**R-BULK.11** - CSV Bulk Data SHOULD be encoded as per [RFC 4180](https://tools.ietf.org/html/rfc4180), MUST contain a header line (column headers), and the media type MUST indicate the presence of the header line.

For example: Content-Type: text/csv; charset=UTF-8; header=present

In addition, the characters used to separate fields and rows as well as identify the escape character can be configured from the characters used in [RFC 4180](https://tools.ietf.org/html/rfc4180).

Using the HTTP example above, the following configures the Agent to transfer data to the Bulk Data Collector using CSV encoding, separating the fields with a comma and the rows with a new line character, by setting the following parameters:

.BulkData.Profile.1.EncodingType = "CSV"  
.BulkData.Profile.1 CSVEncoding.FieldSeparator = ","  
.BulkData.Profile.1.CSVEncoding.RowSeparator="&#13;&#10;"  
.BulkData.Profile.1.CSVEncoding.EscapeCharacter="&quot;"

#### Defining the Report Layout of the Encoded Bulk Data

The layout of the data in the reports associated with the profiles allows parameters to be formatted either as part of a column (ParameterPerColumn) or as a distinct row (ParameterPerRow) as defined below. In addition, the report layout allows rows of data to be inserted with a timestamp stating when the data is collected.

Using the HTTP example above, the following configures the Agent to format the data using a parameter as a row and inserting a timestamp as the first column entry in each row using the "Unix-Epoch" time. The information is configured by setting the following parameters:

.BulkData.Profile.1.CSVEncoding.ReportFormat ="ParameterPerRow"  
.BulkData.Profile.1.CSVEncoding.RowTimestamp ="Unix-Epoch"

The report format of "ParameterPerRow" MUST format each parameter using the ParameterName, ParameterValue and ParameterType in that order. The ParameterType MUST be the parameter's base data type as described in [TR-106](https://www.broadband-forum.org/technical/download/TR-106_Amendment-8.pdf).

#### Layout of Content for Failed Report Transmissions

When the value of the NumberOfRetainedFailedReports parameter of the BulkData.Profile object instance is -1 or greater than 0, then the report of the current reporting interval is appended to the failed reports. For CSV Encoded data the content of new reporting interval is added onto the existing content without any header data.

### Encoding of JSON Bulk Data

Using the HTTP example above, the Set message is used to configure the Agent to transfer data to the Bulk Data Collector using JSON encoding as follows:

.BulkData.Profile.1.EncodingType = "JSON"

#### Defining the Report Layout of the Encoded Bulk Data

Reports that are encoded with JSON Bulk Data are able to utilize different report format(s) defined by the JSONEncoding object’s ReportFormat parameter as defined below.

In addition, a "CollectionTime" JSON object can be inserted into the report instance that defines when the data for the report was collected.

The following configures the Agent to encode the data using a parameter as JSON Object named "CollectionTime" using the "Unix-Epoch" time format:

.BulkData.Profile.1.JSONEncoding.ReportTimestamp ="Unix-Epoch"

Note: The encoding format of "CollectionTime" is defined as an JSON Object parameter encoded as: "CollectionTime":1364529149

Reports are defined as an Array of Report instances encoded as:

"Report":[{...},{...}]

*Note: Multiple instances of Report instances may exist when previous reports have failed to be transmitted.*

#### Layout of Content for Failed Report Transmissions

When the value of the NumberOfRetainedFailedReports parameter of the BulkData.Profile object instance is -1 or greater than 0, then the report of the current reporting interval is appended to the failed reports. For JSON Encoded data the report for the current reporting interval is added onto the existing appended as a new "Data" object array instance as shown below:

"Report": [  
{Report from a failed reporting interval},  
{Report from the current reporting interval}  
]

#### Using the ObjectHierarchy Report Format

When a BulkData profile utilizes the JSON encoding type and has a JSONEncoding.ReportFormat parameter value of "ObjectHierarchy", then the JSON objects are encoded such that each object in the object hierarchy of the data model is encoded as a corresponding hierarchy of JSON Objects with the parameters (i.e., parameterName, parameterValue) of the object specified as name/value pairs of the JSON Object.

For example the translation for the leaf object "Device.MoCA.Interface.\*.Stats." would be:

{  
 "Report": [  
 {  
 "Device": {  
 "MoCA": {  
 "Interface": {  
 "1": {  
 "Stats": {  
 "BroadPktSent": 25248,  
 "BytesReceived": 200543250,  
 "BytesSent": 25248,  
 "MultiPktReceived": 200543250  
 }  
 },  
 "2": {  
 "Stats": {  
 "BroadPktSent": 93247,  
 "BytesReceived": 900543250,  
 "BytesSent": 93247,  
 "MultiPktReceived": 900543250  
 }  
 }  
 }  
 }  
 }  
 }  
 ]  
 }

*Note: The translated JSON Object name does not contain the trailing period "." of the leaf object.*

#### Using the NameValuePair Report Format

When a BulkData profile utilizes the JSON encoding type and has a JSONEncoding.ReportFormat parameter value of "NameValuePair", then the JSON objects are encoded such that each parameter of the data model is encoded as an array instance with the parameterName representing JSON name token and parameterValue as the JSON value token.

For example the translation for the leaf object "Device.MoCA.Interface.\*.Stats." would be:

{  
 "Report": [  
 {  
 "Device.MoCA.Interface.1.Stats.BroadPktSent": 25248,  
 "Device.MoCA.Interface.1.Stats.BytesReceived": 200543250,  
 "Device.MoCA.Interface.1.Stats.BytesSent": 25248,  
 "Device.MoCA.Interface.1.Stats.MultiPktReceived": 200543250,  
 "Device.MoCA.Interface.2.Stats.BroadPktSent": 93247,  
 "Device.MoCA.Interface.2.Stats.BytesReceived": 900543250,  
 "Device.MoCA.Interface.2.Stats.BytesSent": 93247,  
 "Device.MoCA.Interface.2.Stats.MultiPktReceived": 900543250  
 }  
 ]  
}

*Note: The translated JSON Object name does not contain the trailing period "." of the leaf object.*

#### Translating Data Types

JSON has a number of basic data types that are translated from the base data types defined in [TR-106](https://www.broadband-forum.org/technical/download/TR-106_Amendment-8.pdf). The encoding of JSON Data Types MUST adhere to [RFC 7159](https://tools.ietf.org/html/rfc7159).

TR-106 named data types are translated into the underlying base TR-106 data types. Lists based on TR-106 base data types utilize the JSON String data type.

|  |  |
| --- | --- |
| TR-106 Data Type | JSON Data Type |
| base64 | String: base64 representation of the binary data. |
| boolean | Boolean |
| dateTime | String represented as an ISO-8601 timestamp. |
| hexBinary | String: hex representation of the binary data. |
| int, long, unsignedInt, unsignedLong | Number |
| string | String |

## Report Examples

This section provides example report configurations along with the examples of how the resulting encoded data would look as it is transferred to the Bulk Data Collector.

### CSV Encoded Report Examples

#### CSV Encoded Reporting Using ParameterPerRow Report Format

Using the configuration examples provided in the previous sections the configuration for a CSV encoded HTTP report using the ParameterPerRow report format:

.BulkData.Profile.1  
.BulkData.Profile.1.Enable=true  
.BulkData.Profile.1.Protocol = "HTTP"  
.BulkData.Profile.1.ReportingInterval = 300  
.BulkData.Profile.1.TimeReference = "0001-01-01T00:00:00Z"  
.BulkData.Profile.1.HTTP.URL = "https://bdc.acme.com/somedirectory"  
.BulkData.Profile.1.HTTP.Username = "username"  
.BulkData.Profile.1.HTTP.Password = "password"  
.BulkData.Profile.1.HTTP.Compression = "Disabled"  
.BulkData.Profile.1.HTTP.Method = "POST"  
.BulkData.Profile.1.HTTP.UseDateHeader = true  
.BulkData.Profile.1.EncodingType = "CSV"  
.BulkData.Profile.1 CSVEncoding.FieldSeparator = ","  
.BulkData.Profile.1.CSVEncoding.RowSeparator="&#13;&#10;"  
.BulkData.Profile.1.CSVEncoding.EscapeCharacter="&quot;"  
.BulkData.Profile.1.CSVEncoding.ReportFormat ="ParameterPerRow"  
.BulkData.Profile.1.CSVEncoding.ReportTimestamp ="Unix-Epoch"  
.BulkData.Profile.1.Parameter.1.Name = ""  
.BulkData.Profile.1.Parameter.1.Reference = "Device.MoCA.Interface.1.Stats.BroadPktSent"  
.BulkData.Profile.1.Parameter.2.Name = ""  
.BulkData.Profile.1.Parameter.2.Reference = "Device.MoCA.Interface.1.Stats.BytesReceived"  
.BulkData.Profile.1.Parameter.3.Name = ""  
.BulkData.Profile.1.Parameter.3.Reference = "Device.MoCA.Interface.1.Stats.BytesSent"  
.BulkData.Profile.1.Parameter.4.Name = ""  
.BulkData.Profile.1.Parameter.4.Reference = "Device.MoCA.Interface.1.Stats.MultiPktReceived"

The resulting CSV encoded data would look like:

ReportTimestamp,ParameterName,ParameterValue,ParameterType  
1364529149,Device.MoCA.Interface.1.Stats.BroadPktSent,25248,unsignedLong  
1364529149,Device.MoCA.Interface.1.Stats.BytesReceived,200543250,unsignedLong  
1364529149, Device.MoCA.Interface.1.Stats.Stats.BytesSent,7682161,unsignedLong  
1364529149,Device.MoCA.Interface.1.Stats.MultiPktReceived,890682272,unsignedLong

#### CSV Encoded Reporting Using ParameterPerColumn Report Format

Using the configuration examples provided in the previous sections the configuration for a CSV encoded HTTP report using the ParameterPerColumn report format:

.BulkData.Profile.1  
.BulkData.Profile.1.Enable=true  
.BulkData.Profile.1.Protocol = "HTTP"  
.BulkData.Profile.1.ReportingInterval = 300  
.BulkData.Profile.1.TimeReference = "0001-01-01T00:00:00Z"  
.BulkData.Profile.1.HTTP.URL = "https://bdc.acme.com/somedirectory"  
.BulkData.Profile.1.HTTP.Username = "username"  
.BulkData.Profile.1.HTTP.Password = "password"  
.BulkData.Profile.1.HTTP.Compression = "Disabled"  
.BulkData.Profile.1.HTTP.Method = "POST"  
.BulkData.Profile.1.HTTP.UseDateHeader = true  
.BulkData.Profile.1.EncodingType = "CSV"  
.BulkData.Profile.1 CSVEncoding.FieldSeparator = ","  
.BulkData.Profile.1.CSVEncoding.RowSeparator="&#13;&#10;"  
.BulkData.Profile.1.CSVEncoding.EscapeCharacter="&quot;"  
.BulkData.Profile.1.CSVEncoding.ReportFormat ="ParameterPerColumn"  
.BulkData.Profile.1.CSVEncoding.ReportTimestamp ="Unix-Epoch"  
.BulkData.Profile.1.Parameter.1.Name = "BroadPktSent"  
.BulkData.Profile.1.Parameter.1.Reference = "Device.MoCA.Interface.1.Stats.BroadPktSent"  
.BulkData.Profile.1.Parameter.2.Name = "BytesReceived"  
.BulkData.Profile.1.Parameter.2.Reference = "Device.MoCA.Interface.1.Stats.BytesReceived"  
.BulkData.Profile.1.Parameter.3.Name = "BytesSent"  
.BulkData.Profile.1.Parameter.3.Reference = "Device.MoCA.Interface.1.Stats.BytesSent"  
.BulkData.Profile.1.Parameter.4.Name = "MultiPktReceived"  
.BulkData.Profile.1.Parameter.4.Reference = "Device.MoCA.Interface.1.Stats.MultiPktReceived"

The resulting CSV encoded data with transmission of the last 3 reports failed to complete would look like:

ReportTimestamp,BroadPktSent,BytesReceived,BytesSent,MultiPktReceived  
1364529149,25248,200543250,7682161,890682272  
1464639150,25249,200553250,7683161,900683272  
1564749151,25255,200559350,7684133,910682272  
1664859152,25252,200653267,7685167,9705982277

### JSON Encoded Report Example

Using the configuration examples provided in the previous sections the configuration for a JSON encoded HTTP report:

.BulkData.Profile.1  
.BulkData.Profile.1.Enable=true  
.BulkData.Profile.1.Protocol = "HTTP"  
.BulkData.Profile.1.ReportingInterval = 300  
.BulkData.Profile.1.TimeReference = "0001-01-01T00:00:00Z"  
.BulkData.Profile.1.HTTP.URL = "https://bdc.acme.com/somedirectory"  
.BulkData.Profile.1.HTTP.Username = "username"  
.BulkData.Profile.1.HTTP.Password = "password"  
.BulkData.Profile.1.HTTP.Compression = "Disabled"  
.BulkData.Profile.1.HTTP.Method = "POST"  
.BulkData.Profile.1.HTTP.UseDateHeader = true  
.BulkData.Profile.1.EncodingType = "JSON"  
.BulkData.Profile.1.JSONEncoding.ReportFormat ="ObjectHierarchy"  
.BulkData.Profile.1.JSONEncoding.ReportTimestamp ="Unix-Epoch"  
.BulkData.Profile.1.Parameter.1.Reference = "Device.MoCA.Interface.\*.Stats."

The resulting JSON encoded data would look like:

{  
 "Report": [  
 {  
 "CollectionTime": 1364529149,  
 "Device": {  
 "MoCA": {  
 "Interface": {  
 "1": {  
 "Stats": {  
 "BroadPktSent": 25248,  
 "BytesReceived": 200543250,  
 "BytesSent": 25248,  
 "MultiPktReceived": 200543250  
 }  
 },  
 "2": {  
 "Stats": {  
 "BroadPktSent": 93247,  
 "BytesReceived": 900543250,  
 "BytesSent": 93247,  
 "MultiPktReceived": 900543250  
 }  
 }  
 }  
 }  
 }  
 }  
 ]  
}

If the value of the .BulkData.Profile.1.JSONEncoding.ReportFormat parameter was "NameValuePair", the results of the configuration would be:

{  
 "Report": [  
 {  
 "CollectionTime": 1364529149,  
 "Device.MoCA.Interface.1.Stats.BroadPktSent": 25248,  
 "Device.MoCA.Interface.1.Stats.BytesReceived": 200543250,  
 "Device.MoCA.Interface.1.Stats.BytesSent": 25248,  
 "Device.MoCA.Interface.1.Stats.MultiPktReceived": 200543250,  
 "Device.MoCA.Interface.2.Stats.BroadPktSent": 93247,  
 "Device.MoCA.Interface.2.Stats.BytesReceived": 900543250,  
 "Device.MoCA.Interface.2.Stats.BytesSent": 93247,  
 "Device.MoCA.Interface.2.Stats.MultiPktReceived": 900543250  
 }  
 ]  
}