Ember+ Matrix Representation

Proposal for an extension to the Glow Schema used by the Ember+ protocol

Author: pbo@l-s-b.de

Date: 2012-08-29
Revision: 2

Contents

Revision History

Revision	Date	Changes
2	2012-08-29	• Signal numbers in linear addressing mode are now zero-based instead of one-based. See type Signal. • The properties `Matrix.targets` and `Matrix.sources` may be omitted for matrices with linear addressing. See type Matrix. • A provider must report connections for all targets. Omitting targets with no active connections is no longer allowed. See type Matrix. • The `Connection.sources` property is now optional. See type Connection.
1	2012-08-27	• Initial Release

Introduction

This document describes a possible way of integrating matrices into the Glow schema. A matrix in the sense of this document is a two-dimensional array of Boolean values.

Possible applications of matrices in the context of gadget control:

- Signal routing
- GP-I/O signaling
- Key assignment for Intercom systems
- Group and conference management for Intercom systems

The entities represented by the rows and columns of a matrix are called signals. The signals represented by the matrix columns are called targets. The signals represented by the matrix rows are called sources. The values contained in the matrix cells may be interpreted as connections from sources to targets: If `matrix[S, T]` is true, source S is connected to target T.

Matrices inherently have three different connection semantics:

- 1:N
  one source may be connected to n targets, but each target must not be connected to more than one source
- 1:1
  one source may be connected to only one target, and each target must not be connected to more than one source
- N:N
  a source may be connected to n targets, and a target may have n sources connected to it.

Ember+ must support all three matrix types.

For N:N matrices, connections are usually described by multiple values – not only the Boolean state of the connection, but also a ratio or weight for each connected source. This proposal takes this into account by allowing a matrix to refer to a collection of Parameters describing each signal and connection.

Also, the signals making up a matrix may have multiple labels, each of which may be editable. This proposal takes this into account by allowing a matrix to refer to multiple collections of Parameters which describe signal labels.

Type Definitions

ASN.1 Notation

The new type Matrix stands alongside the types Node and Parameter (defined in Glow Schema 2.5):

```
Matrix ::=
    [APPLICATION 13] IMPLICIT
        SEQUENCE {
            number          [0] Integer32,
            contents        [1] MatrixContents       OPTIONAL,
            children        [2] ElementCollection    OPTIONAL,
            targets         [3] TargetCollection     OPTIONAL,
            sources         [4] SourceCollection     OPTIONAL,
            connections     [5] ConnectionCollection OPTIONAL
        }


MatrixContents ::=
    SET {
        identifier                [ 0] EmberString,
        description               [ 1] EmberString         OPTIONAL,
        type                      [ 2] MatrixType          OPTIONAL,
        addressingMode            [ 3] MatrixAddressingMode OPTIONAL,
        targetCount               [ 4] Integer32,
        sourceCount               [ 5] Integer32,
        maximumTotalConnects      [ 6] Integer32           OPTIONAL,
        maximumConnectsPerTarget  [ 7] Integer32           OPTIONAL,
        parametersLocation        [ 8] ParametersLocation  OPTIONAL,
        gainParameterNumber       [ 9] Integer32           OPTIONAL,
        labels                    [10] LabelCollection     OPTIONAL
    }


MatrixType ::=
    INTEGER {
        oneToN   (0),
        oneToOne (1),
        nToN     (2)
    }


MatrixAddressingMode ::=
    INTEGER {
        linear    (0),
        nonLinear (1)
    }
```

```
ParametersLocation ::=
    CHOICE {
        basePath RELATIVE-OID,
        inline   Integer32
    }


LabelCollection ::=
    SEQUENCE OF [0] Label


Label ::=
    [APPLICATION 18] IMPLICIT
        SEQUENCE {
            basePath    [0] RELATIVE-OID,
            description [1] EmberString
        }


TargetCollection ::=
    SEQUENCE OF [0] Target


Target ::=
    [APPLICATION 14] IMPLICIT
        Signal


Signal ::=
    SEQUENCE {
        number  [0] Integer32,
    }


SourceCollection ::=
    SEQUENCE OF [0] Source


Source ::=
    [APPLICATION 15] IMPLICIT
        Signal


ConnectionCollection ::=
    SEQUENCE OF [0] Connection


Connection ::=
    [APPLICATION 16] IMPLICIT
        SEQUENCE {
            target         [0] Integer32,
            sources          [1] PackedNumbers        OPTIONAL,
            operation        [2] ConnectionOperation  OPTIONAL,
            errorLevel       [3] ConnectionErrorLevel OPTIONAL,
        }


PackedNumbers ::=
    RELATIVE-OID


ConnectionOperation ::=
    INTEGER {
        absolute   (0),
        connect    (1),
        disconnect (2)
    }


ConnectionErrorLevel ::=
    INTEGER {
        tally    (0),
        taken    (1),
        pending  (2),
        locked   (3)
        -- more tbd.
    }


QualifiedMatrix ::=
    [APPLICATION 17] IMPLICIT
        SEQUENCE {
            path             [0] RELATIVE-OID,
            contents         [1] MatrixContents       OPTIONAL,
            children         [2] ElementCollection    OPTIONAL,
            targets          [3] TargetCollection     OPTIONAL,
            sources          [4] SourceCollection     OPTIONAL,
            connections      [5] ConnectionCollection OPTIONAL
        }
```

Matrix

- `number [0] Integer32`
  The number of the matrix object. Has same semantics as `Parameter.number` and `Node.number`.
- `contents [1] MatrixContents OPTIONAL`
  Contents set of the matrix object. Analogical to `Parameter.contents` and `Node.contents`.
- `children [2] ElementCollection OPTIONAL`

Contains child elements of the matrix object. Conceptually, matrix objects are considered leafs in the Glow tree. Though, a matrix may contain a Glow Command in the same way as a parameter.

- `targets [3] TargetCollection OPTIONAL`
  A collection of `Target` objects that fill columns of the matrix. This property must only be present if the matrix addressing mode is non-linear. A linear matrix may use this property to restrict connections to the targets listed under this property.
- `sources [4] SourceCollection OPTIONAL`
  A collection of `Source` objects that fill rows of the matrix. This property must only be present if the matrix addressing mode is non-linear. A linear matrix may use this property to restrict connections to the sources listed under this property.
- `connections [5] ConnectionCollection OPTIONAL`
  A collection of `Connection` objects that describe each target's active connection. The provider must report one `Connection` object for each target.

When the children of a node are requested (using the `GetDirectory` command) and this node contains a Matrix object, the matrix is transmitted only containing the properties `number` and `contents`.
The consumer may then issue a `GetDirectory` command as a child of the matrix object. The provider must reply with the complete matrix object, containing at least the property `connections.` In the case of a matrix with non-linear addressing mode, the `targets` and `sources` properties are also required.

As soon as a consumer issues a `GetDirectory` command on a matrix object, it implicitly subscribes to matrix connection changes. The consumer may afterwards use the `Unsubscribe` command to cancel this subscription.

MatrixContents

- `identifier [0] EmberString`
  The object identifier. Analogical to `Parameter.identifier` and `Node.identifier`.
- `description [1] EmberString OPTIONAL`
  The display name of the matrix object. Analogical to `Parameter.description` and `Node.description`.
- `type [2] MatrixType OPTIONAL`
  The type of the matrix: either 1:N, 1:1 or N:N. If this property is not present, 1:N is assumed.
- `addressingMode [3] MatrixAddressingMode OPTIONAL`
  The addressing mode used by the matrix. Either linear (signal numbers are row/column indices) or non-linear (signal numbers are random).
- `targetCount [4] Integer32`
  If `addressingMode` is linear, this defines the number of matrix columns. If `addressingMode` is non-linear, this defines the number of targets contained in the targets property.
- `sourceCount [5] Integer32`
  If `addressingMode` is linear, this defines the number of matrix rows. If `addressingMode` is non-linear, this defines the number of sources contained in the sources property.
- `maximumTotalConnects [6] Integer32 OPTIONAL`

This property is only significant for N:N matrices. It defines the maximum number of active connections for the entire matrix. If this property is not present, `targetCount` * `sourceCount` is assumed.

- `maximumConnectsPerTarget [7] Integer32 OPTIONAL`
  This property is only significant for N:N matrices. It defines the maximum number of sources that may be connected to one target. If this property is not present, `sourceCount` is assumed.
- `parametersLocation [8] ParametersLocation OPTIONAL`
  This property may refer to the location of parameters associated with the signals and connections of the matrix.
- `gainParameterNumber [9] Integer32 OPTIONAL`
  This property is only significant for N:N matrices and may only be present if the `parametersLocation` property is also present. It may contain the number of the parameter that describes the gain or ratio of a connection. This parameter must exist for every connection under the location referred to by the `parametersLocation` property. It must be either of type INTEGER or of type REAL.
- `labels [10] LabelCollection OPTIONAL`
  This property may contain multiple references to Nodes which contain signal label parameters. These parameters must be of type `EmberString`.

MatrixType

- `oneToN (0)`
  Default. The matrix uses 1:N connection semantics.
- `oneToOne (1)`
  The matrix uses 1:1 connection semantics.
- `nToN (2)`
  The matrix uses N:N connection semantics.

MatrixAddressingMode

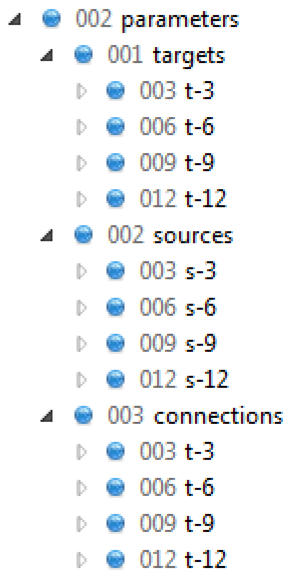
- `linear (0)`
  Default. The matrix uses linear addressing mode.
- `nonLinear (1)`
  The matrix uses non-linear addressing mode.

ParametersLocation

- `basePath RELATIVE-OID`
  Contains an absolute path to a node under which parameters associated with signals or connections reside.
- `inline Integer32`

Contains a single sub-identifier to be appended to the path of the matrix. The resulting path refers to the node under which parameters associated with signals or connections reside. This node is not reported in a response to a `GetDirectory` command, since huge matrices with hundreds or thousands of connections will not be able to publish all connection and signal parameters in the Glow tree. The provider must create the parameter objects on-the-fly when a `GetDirectory` command is issued to a sub-node containing the parameters for a specific signal or connection.

The sub-tree under the node a `ParametersLocation` object refers to must be structured like this:



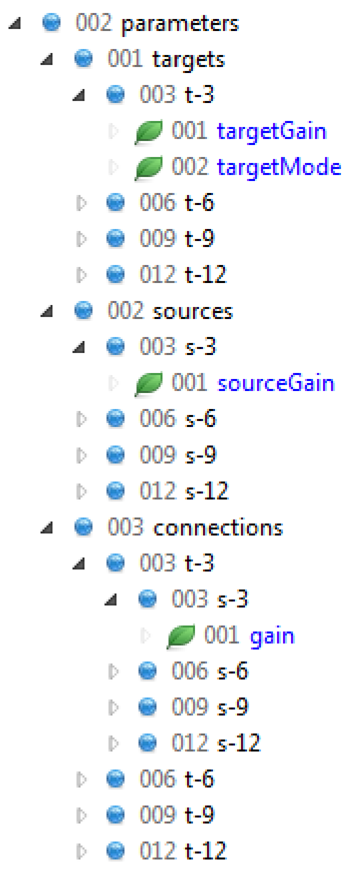
```
002 parameters
  001 targets
    003 t-3
    006 t-6
    009 t-9
    012 t-12
  002 sources
    003 s-3
    006 s-6
    009 s-9
    012 s-12
  003 connections
    003 t-3
    006 t-6
    009 t-9
    012 t-12
```

In this sample, a `ParametersLocation` object refers to the node "`2 - parameters`". This node must contain three well-known nodes: "`1 – targets`", "`2 – sources`" and "`3 – connections`".

The node "`1 – targets`" must contain one node for each target associated with parameters. Each of these nodes must have the number of the corresponding target.

The node "`2 – sources`" must contain one node for each source associated with parameters. Each of these nodes must have the number of the corresponding source.

The node "`3 – connections`" must contain one node for each target. These target nodes must themselves contain one node for each source:



```
002 parameters
  001 targets
    003 t-3
      001 targetGain
      002 targetMode
    006 t-6
    009 t-9
    012 t-12
  002 sources
    003 s-3
      001 sourceGain
    006 s-6
    009 s-9
    012 s-12
  003 connections
    003 t-3
      003 s-3
        001 gain
      006 s-6
      009 s-9
      012 s-12
    006 t-6
    009 t-9
    012 t-12
```

To request all parameters associated with target 3, the consumer may issue a `GetDirectory` command contained in a `QualifiedNode` object with the path "`<parametersLocation>.1.3`".

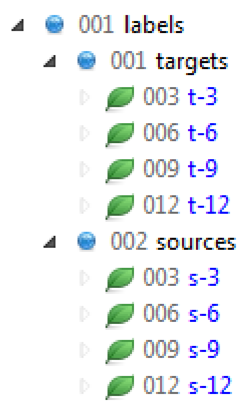
To request all parameters associated with source 6, the consumer may issue a `GetDirectory` command contained in a `QualifiedNode` object with the path "`<parametersLocation>.2.6`".

To request all parameters associated with connection "target 9 <- source 12", the consumer may issue a GetDirectory command contained in a `QualifiedNode` object with the path "`<parametersLocation>.3.9.12`".

Label

- `basePath [0] RELATIVE-OID`
  This property contains an absolute path to a node under which label parameters associated with signals reside.
- `description [1] EmberString`
  This property contains a free-text description of the label, which may be used by consumers to let the user choose the label to display.

The sub-tree under the node a `Label.basePath` refers to must be structured like this:



```
001 labels
    001 targets
        003 t-3
        006 t-6
        009 t-9
        012 t-12
    002 sources
        003 s-3
        006 s-6
        009 s-9
        012 s-12
```

In this sample, a `Label.basePath` refers to the node "`1 - labels`". This node must contain two well-known nodes: "`1 – targets`" and "`2 – sources`".

The node "`1 – targets`" must contain one parameter of type `EmberString` for each target associated with a label. Each of these parameters must have the number of the corresponding target.

The node "`2 – sources`" must contain one parameter of type `EmberString` for each source associated with a label. Each of these parameters must have the number of the corresponding source.

Signal

- `number [0] Integer32`
  This property contains the number of the signal.
  If the addressing mode of the encompassing matrix is linear, this number is the zero-based index of the row or column containing the signal.
  If the addressing mode of the encompassing matrix is non-linear, this is a random number uniquely identifying the signal.

Connection

- `target [0] Integer32`
  This property contains the number of the connection target.
- `sources [1] PackedNumbers OPTIONAL`
  This property contains the numbers of all sources connected to `target`. If not present, the target does not have any active connections.
- `operation [2] ConnectionOperation OPTIONAL`
  This property may contain a hint on how the set of sources must be interpreted. If this property is not present, "absolute" is assumed.
- `errorLevel [3] ConnectionErrorLevel OPTIONAL`
  When the connection object is sent by the provider as the response to a connect request, this property may contain an error code indicating the execution state of the operation. If this property is not present, "tally" is assumed.

The consumer should not transmit this property at all when issuing connect requests.

PackedNumbers

This type is an alias for the primitive ASN.1 type RELATIVE-OID. It is used to hold a vector of integer values in a compact way.

ConnectionOperation

- `absolute (0)`
  Default. This value indicates that the list of source numbers in the sources property of the encompassing Connection object is absolute. When a provider tallies a connection, it must always use this value, independent of the value of the errorLevel property.
- `connect (1)`
  This indicates that the sources in the sources property of the encompassing Connection object should be connected to the given target. Only used by the consumer when issuing a connect request.
- `disconnect (2)`
  This indicates that the sources in the sources property of the encompassing Connection object should be disconnected from the given target. Only used by the consumer when issuing a connect request.

ConnectionErrorLevel

- `tally (0)`
  Default. The operation property of the encompassing Connection object must be set to “absolute” and the sources property must contain the absolute set of sources currently connected.
- `taken (1)`
  May be used by the provider to indicate success of a connect operation issued by the consumer. The sources property of the encompassing Connection must contain the absolute set of sources currently connected.
- `pending (2)`
  May be used by the provider to signal that the connect operation was queued, but is not yet current.
- `locked (3)`
  May be used by the provider to signal that the connect operation could not be executed because the target is locked. The sources property of the encompassing Connection must contain the absolute set of sources currently connected.

QualifiedMatrix

- `path [0] RELATIVE-OID`
  The absolute path to the Matrix. Uses the same semantics as `QualifiedParameter.path` and `QualifiedNode.path`.
- `contents [1] MatrixContents OPTIONAL`
  See `Matrix.contents`.
- `children [2] ElementCollection OPTIONAL`
  See `Matrix.children`.
- `targets [3] TargetCollection OPTIONAL`
  See `Matrix.targets`.
- `sources [4] SourceCollection OPTIONAL`
  See `Matrix.sources`.
- `connections [5] ConnectionCollection OPTIONAL`
  See `Matrix.connections`.

Changes to Existing Types

The following changes only add to the existing types to preserve backwards compatibility.

```
Element ::=
    CHOICE {
        parameter          Parameter,
        node               Node,
        command            Command,
        matrix             Matrix
    }


FieldFlags ::=
    INTEGER {
        all               (-1),
        default           ( 0), -- same as "all"
        identifier        ( 1),
        description       ( 2),
        tree              ( 3),
        value             ( 4),
        connections       ( 5)
    }


RootElement ::=
    CHOICE {
        element            Element,
        qualifiedParameter QualifiedParameter,
        qualifiedNode      QualifiedNode,
        qualifiedMatrix    QualifiedMatrix
    }
```

Element

This change ensures that objects of type `Matrix` can be added to the Glow tree.

FieldFlags

- `connections (8)`
  When issuing a `GetDirectory` command directly on a Matrix object, the consumer may specify this value to indicate that only the matrix connections property should be returned.

RootElement

This change ensures that objects of type `QualifiedMatrix` can be added directly below the root level of the Glow tree.

Use Cases

The following use cases are illustrated as XML samples (Glow is easily transposed to XML).

Transmission of the Matrix Description

Consumer -> Provider:

```
<QualifiedNode path="1.2">
  <children>
    <Command number="32" />
  </children>
</QualifiedNode>
```

This message requests all children under the node with path “1.2”.

Provider -> Consumer:

```
<QualifiedNode number="1.2">
  <children>
    <Matrix number="1">
      <contents>
        <identifier>matrix</identifier>
        <description>Sample Matrix</description>
        <targetCount>4</targetCount>
        <sourceCount>4</sourceCount>
        <type>nToN</type>
        <parametersLocation type="RelativeOid">1.2.2</parametersLocation>
        <gainParameterNumber>1</gainParameterNumber >
        <labels>
          <Label basePath="1.2.3.1" description="Primary" />
          <Label basePath="1.2.3.2" description="Internal" />
        </labels>
      </contents>
    </Matrix>
  </children>
</QualifiedNode>
```

This message reports one child under the requested node: a matrix of type N:N, with linear addressing mode, 4 targets, 4 sources, a reference to associated parameters and references to two layers of labels.

Transmission of Connections

Consumer -> Provider:

```
<QualifiedMatrix path="1.2.1">
  <children>
    <Command number="32" />
  </children>
</QualifiedMatrix>
```

This message requests all sub-elements of the matrix (targets, sources and connections).

Provider -> Consumer:
```
<QualifiedMatrix path="1.2.1">
  <contents>
    <identifier>matrix</identifier>
    <!-- … -->
  </contents>
  <connections>
    <Connection target="0">
      <sources>3</sources>
    </Connection>
    <Connection target="1">
      <sources>0.1</sources>
    </Connection>
    <Connection target="2">
      <sources>3.1.2</sources>
    </Connection>
    <Connection target="3" />
  </connections
</QualifiedMatrix>
```
This message includes the matrix contents and connections. Targets and sources are omitted since the matrix uses linear addressing mode.

Issuing Connects

Consumer -> Provider:
```
<QualifiedMatrix path="1.2.1">
  <connections>
    <Connection target="0">
      <sources>0.2</sources>
      <operation>connect</operation>
    </Connection>
    <Connection target="1">
      <sources>2</sources>
      <operation>connect</operation>
    </Connection>
  </connections>
</QualifiedMatrix>
```
This message requests a connection to be made from sources 0 and 2 to target 0 and a connection to be made from source 2 to target 1.

Provider -> Consumer:
```
<QualifiedMatrix path="1.2.1">
  <connections>
    <Connection target="0">
      <sources>3.0.2</sources>
      <errorLevel>taken</errorLevel>
    </Connection>
    <Connection target="1">
      <sources>0.2</sources>
      <errorLevel>taken</errorLevel>
    </Connection>
  </connections>
</QualifiedMatrix>
```

This message reports the new state of the connections to targets 0 and target 1.

Issuing Disconnects

Consumer -> Provider:

```
<QualifiedMatrix path="1.2.1">
  <connections>
    <Connection target="0">
      <sources>3</sources>
      <operation>disconnect</operation>
    </Connection>
  </connections>
</QualifiedMatrix>
```

This message requests source 3 to be disconnected from target 0.

Provider -> Consumer:

```
<QualifiedMatrix path="1.2.1">
  <connections>
    <Connection target="0">
      <sources>0.2</sources>
      <errorLevel>taken</errorLevel>
    </Connection>
  </connections>
</QualifiedMatrix>
```

This message reports the new state of the connections to target 0.

Requesting Labels

Consumer -> Provider:

```
<QualifiedNode path="1.2.3.1.1">
  <!-- path: 1.2.3.1       .1
             label.basePath targets -->
  <children>
    <Command number="32" dirFieldMask="4" />
  </children>
</QualifiedNode>
```

This is a usual `GetDirectory` message issued on the first layer of target labels reported by the matrix. It also asks the provider to only return the values of the found parameters.

Provider -> Consumer:

```
<QualifiedNode path="1.2.3.1.1">
  <children>
    <Parameter number="0">
      <contents>
        <value type="UTF8">Primary Label of Target 0</value>
      </contents>
    </Parameter>
    <Parameter number="1">
      <contents>
        <value type="UTF8">Primary Label of Target 1</value>
      </contents>
    </Parameter>
    <Parameter number="2">
      <contents>
        <value type="UTF8">Primary Label of Target 2</value>
      </contents>
    </Parameter>
    <Parameter number="3">
      <contents>
        <value type="UTF8">Primary Label of Target 3</value>
      </contents>
    </Parameter>
  </children>
</QualifiedNode>
```

This response includes all label parameters. Note that each parameter's number equals the number of the target it is associated with.

Changing Labels

Consumer -> Provider:

```
<QualifiedParameter path="1.2.3.1.1.3">
  <!-- path: 1.2.3.1       .1      .3
             label.basePath targets target3 -->
  <contents>
    <value type="UTF8">New Primary Label of Target 3</value>
  </contents>
</QualifiedNode>
```

This is the standard procedure for changing a parameter.

Provider -> Consumer:

```
<QualifiedParameter path="1.2.3.1.1.3">
  <contents>
    <value type="UTF8">New Primary Label of Target 3</value>
  </contents>
</QualifiedNode>
```

And this is the response signaling success.

Requesting Connection Parameters

Consumer -> Provider:

```
<QualifiedNode path="1.2.2.3.0.3">
  <!-- path: 1.2.2              .3           .0      .3
            parametersLocation connections target0 source3 -->
  <children>
    <Command number="32" />
  </children>
</QualifiedNode>
```

This is a usual `GetDirectory` message issued on the node including the parameters for connection `target0 <- source3.`

Provider -> Consumer:

```
<QualifiedNode path="1.2.2.3.0.3">
  <children>
    <Parameter number="1">
      <contents>
        <identifier>gain</identifier>
        <description>gain of connection target0 <- source3</description>
        <value type="Real">-64.0</value>
        <minimum type="Real">-128.0</value>
        <maximum type="Real">15.0</maximum>
        <access>readWrite</access>
      </contents>
    </Parameter>
    <Parameter number="2">
      <contents>
        <identifier>peak</identifier>
        <description>peak level of connection target0 <-
source3</description>
        <type>Integer</type>
        <minimum type="Real">0</value>
        <maximum type="Real">255</maximum>
        <streamIdentifier>110</streamIdentifier>
      </contents>
    </Parameter>
  </children>
</QualifiedNode>
```

This response includes two parameters: a gain parameter and a streaming-enabled peak parameter.

Changing Connection Parameters

Consumer -> Provider:

```
<QualifiedParameter path="1.2.2.3.0.3.1">
  <!-- path: 1.2.2       .3          .0      .3      .1
       parametersLocation connections target1 source4 gain -->
  <contents>
    <value type="Real">10.0</value>
  </contents>
</QualifiedParameter>
```

This is the standard procedure for changing a parameter.

Provider -> Consumer:

```
<QualifiedParameter path="1.2.2.3.0.3.1">
  <contents>
    <value type="Real">10.0</value>
  </contents>
</QualifiedParameter>
```

And this is the response signaling success.

Performance Characteristics

The following section lists common use cases and the number of bytes needed for transmission. All byte lengths take into account framing and escaping.

- Response to `GetDirectory` command on 1:N matrix 200x200 (transmits matrix contents, 200 targets, 200 sources, 200 connections): **6761 Bytes**.
- Setting a single connection on same matrix (transmits 1 connection): **46 Bytes**.
- Reporting a single connection with `errorLevel “taken”` on same matrix (transmits 1 connection): **51 Bytes**.
- Response to `GetDirectory` command on N:N matrix 4x4 with 4 active connections (transmits matrix contents, 4 targets, 4 sources, 4 connections): **247 Bytes**
- Response to `GetDirectory` command on same matrix but with 16 connections: **259 Bytes**.
- Response to `GetDirectory` command on N:N matrix 1000x1000 with 1000 active connections: **36517 Bytes**.
- Same response as above, but only connections (`GetDirectory` command refined with `dirFieldMask`): **16211 Bytes**
- Response to `GetDirectory` command on same matrix but with hypothetical maximum of 1000000 (one million) active connections: **2025838 Bytes** (1.93 Mbytes)
- Reporting a connection of one target to 1000 sources on same matrix (transmits one connection): **2051 Bytes**.