NDL 1.0 – Network Design Language™

By:

John T. Chapman
Cisco Fellow
jchapman@cisco.com

June 24, 2008

SCTE Cable-Tec Expo® 2008

Table of Contents

List of Figures

Release: 080520
Source Filename: NDL-080520.doc
Document Control: EDCS-668992

Introduction

Network Design Language™, or NDL, is a short hand, algebraic-based notation for describing a network entity such as a router, a cable modem termination system (CMTS), or any other network component. NDL is designed to simplify language complexity and provide a simple framework to represent networks and components.

A representation is devised to decompose complex systems into more tractable subunits, as well as to assemble and map those defined subunits onto actual sentences/formulas that represent the network or component. The intent is to enable anyone involved in any aspect of network design or deployment to easily represent and understand complex systems.

NDL is structured as a natural language. Its primary usage is targeted at "pen and paper", where simpler is better, and where a description of a system component should be concise, and most of all, readable. This is known as informal NDL. There is also a formal version of NDL which has additional structure to allow it to be machine readable. Thus, network descriptions could be written in NDL and then software could compile NDL and produce network diagrams.

NDL has its roots in the cable industry. The author developed NDL as he tried to sort out the twisted interconnectivity of modular CMTSs (M-CMTSs), fiber nodes, and DOCSIS 3.0 service groups, along with asymmetrical downstream (DS) and upstream (US) directions. However, NDL is broadly applicable to any networking scenario.

NDL assembles larger objects out of smaller objects. NDL can start with various line card definitions, and then add them together to build and describe an entire network. Objects have ports. Attributes can then be assigned to the object, the ports, or both.

This paper describes the basic approach of NDL and explains how certain special cases are handled. It provides case studies as examples, followed by formal language descriptions.

Basic Description

NDL describes all network entities through the use of three NDL components:

- Objects
- Ports
- Attributes

The general NDL naming syntax is:

Object { Port List } [Attribute List]

Objects

NDL treats network components as objects. A base-level NDL object is intended to be some sort of field replaceable unit (FRU) such as a line card in a chassis. Examples include routers, switches, and CMTSs, but may also include transmission elements such as splitters or fiber nodes.

In NDL, one of the basic building blocks is a line card, also referred to as LC. So, using the informal version of NDL with no port list, a line card would be shown as:

LC

Now, let's build a generic chassis that has eight line cards. NDL uses the reserved expression of CH for chassis.

CH = 8 * LC

An algebraic expression is used to indicate that eight line cards equal one chassis. The next sections will show that if the equation contained the properties of the line card, the properties of the chassis could be calculated from this equation.

This example illustrates how objects are hierarchical. That is, objects can be added or multiplied to make bigger objects. As the level of hierarchy continues, the objects become more abstract in nature, but still represent a physical collection of network entities. Examples include:

- a service group, which is a collection of frequencies on a set of fiber nodes;
- all the equipment in one rack which may be operating as a unit;
- all the equipment in an entire network.

Ports

These objects can be described by their physical connectivity which is based upon their physical ports. Physical ports generally are ports that occupy space somewhere on a front or back panel and have a cable connected to them. Examples include:

- Gigabit Ethernet (GE) port
- Hybrid Fiber Coaxial (HFC) DS or US port

Port Lists are contained in curly brackets { } immediately following the object name. There are two variables used in describing a port. There is the port type which indicates the function of the port, and the port count which indicates how many of those ports exist for that object.

Let's change our earlier example from a generic chassis into an Ethernet switch that has 24 GE ports on each line card. The line card definition would be:

```
Ethernet_LC { 24 GE }
```

The port type "GE" is a reserved port type which refers to Gigabit Ethernet.

Now, let's use NDL to calculate how many ports exist at the chassis level. The generic chassis expression, "CH", is replaced by a more relevant name.

```
Switch { }
        = 8 * Ethernet_LC { 24 GE }
        = Switch { 8 * 24 GE }
        = Switch { 192 GE }
```

The equation starts declaring that a switch is being defined, but the port types and counts are not known. An algebraic equation is used to indicate that there are eight line cards with 24 Gigabit Ethernet ports per line card. The result is a switch with 192 Gigabit Ethernet ports.

Also note the formatting used. The original declaration is placed on the first line. The subsequent equal signs are lined up for readability, while multiplications are done in line. The final result is on the last line. This style of formatting is chosen for readability. Even though everything could fit into one line, it is much more readable if the lines are short and repeated, or if like objects are lined up in columns rather than spread out in rows.

Attributes

Each object or port may contain one or more attributes. Attributes are uniquely identified by their units. Some examples of attributes include:

- 10 Gbps
- SFP
- 6 MHz
- 500 HHP
- Channelization

Attribute lists are contained in square brackets [ ] immediately following a port or object definition. For example, to indicate the data capacity of an Ethernet line card,

LC { 24 GE [1 Gbps] }
= LC { 24 GE } [24 * 1 Gbps]
= LC { 24 GE } [24 Gbps]

In the first line of this example, the 1 Gbps attribute is attached to the port type, and therefore, is interpreted as a per port attribute. In the second line, a mathematical expression is shown in the attribute list. In the third line, the final result is shown. Since the attribute list on the second and third line is after the closing curly bracket }, the attribute list is considered as being attached to the object.

Any number of attributes could be attached to an equation. As a result, care has to be taken to not add so many attributes that readability is sacrificed. In general, only include attributes that are relevant to the equation. The only time all attributes may be included is in a full prototype of an object.

Also, obvious attributes should be excluded. In this case, the 1 Gbps attribute is obvious because the port type is GE which is known to imply 1 Gbps. It is only included because it makes the math operation more clear. The second line is obvious and could be excluded. The attribute list on the third line is actually useful as it shows the total port capacity of the switch.

There are many ways to have done the formatting. For readability, it works best for the port list to stand out. Thus, the curly brackets for the Port List have one white space immediately inside each one, while the square brackets for the Attribute List have one white space outside them with none immediately inside them. There is always a single white space after a comma. Double spaces should not be used.

Special Cases

Asymmetrical Ports

Most network ports such as Ethernet are bi-directional and support the same data rate in either direction. Sometimes, a wide area network interface such as the serial interface V.35 will have a different egress data rate than ingress data rate. This can be handled through the attribute list.

Serial_LC { 4 Serial } [V.35, 1 Mbps x 256 kbps] }
Serial_LC { 4 Serial } [V.35, 1 Mbps/port x 256 kbps/port]
Serial_LC { 4 Serial } [V.35, 4 Mbps x 1 Mbps]

These examples show different ways of describing the same assembly which has four satellite ports, each of which is 1 Mbps egress and 256 kbps ingress.

Note the use of the "/port" attribute modifier in the Attribute List. This permits port attributes to be listed in the object level attribute list. This is a recommended practice as it keeps the Port List simple and readable.

Uni-directional Ports

Some devices have a separate port for each direction of traffic. A CMTS is a classic example, as not only are the ports unidirectional, but they are asymmetrical in both quantity and capacity.

Background

A CMTS resides at a cable operator's hub or headend site and connects over the HFC plant to a cable modem (CM) in a subscriber's home.

The coax in the subscriber's home is a two-way media based upon frequency division multiplexing (FDM) where, in North America, 5 to 42 MHz is referred to as the upstream spectrum and 54 MHz to 1 GHz is referred to as the downstream spectrum. Whenever these spectrums are passed through active electronics—such as amplifiers—a diplexer is used to separate the two spectrums on to dedicated signal paths so that unidirectional amplification can be used for each.

A CM contains a diplexer whereas a CMTS does not. Thus, a CMTS has separate downstream ports and upstream ports. Further, a CMTS tends to have a different number of downstream and upstream ports to accommodate different fiber node connectivity scenarios

NDL Port Syntax

NDL manages port syntax with two special port types and a special port operator called the NDL "by modifier".

A downstream port has a port type of DS. Downstream is defined as from the headend or hub to the subscriber. In more common network terminology, the downstream is from the provider edge (PE) to the consumer edge (CE). As a result, the DS port is an output port on a CMTS, but an input port to a CM (after the diplexer).

An upstream port has the port type of US. Upstream is defined as from the subscriber to the cable operator's hub or headend. In more common network terminology, the upstream is from CE to PE. As a result, the US is an output from the CM electronics (before the diplexer) and an input to the CMTS.

For example, one type of CMTS line card is the MC520H which has five downstream ports and 20 upstream ports. It would be referred to as:

MC520H { 5 DS, 20 US }

NDL has a short-hand notation for this configuration known as the "by modifier". The "by modifier" is denoted by a lowercase "x". The use of the "by modifier" infers DS x US. The "by modifier" may be used in the port list or in the attribute list. Thus, another way to refer to this line card is:

MC520H { 5 x 20 }

Note that when the "by modifier" is used, the DS and US port types are not used. Also note that the "by modifier" is preceded by one white space and followed by one white space.

Channelization

Ports may also be channelized. To continue with the CMTS example, in a CMTS, channelization is achieved by stacking multiple carrier frequencies on the same port. Channelization can take place in either DS or US directions. DOCSIS also defines logical channels. A logical channel is a further level of sub-channelization within an upstream channel.

NDL treats channels as attributes, not as ports. Channels are attributes of ports. The attribute unit for channels is "ch". When there is only one channel per port, the attribute should not be used unless it is to provide clarity in a calculation.

The MC520H has an operational mode where it can frequency stack two upstream channels into one port. In that mode, only ten upstream ports are available. That can be shown in NDL as follows:

MC520H { 5 x 20 }
= MC520H { 5 x 10 [2 ch]}
= MC520H { 5 x 10 } [5 ch x 20 ch]

or

MC520H { 5 DS, 20 US}
= MC520H { 5 DS, 10 US [2 ch] }
= MC520H { 5 DS, 10 US } [20 US ch]
= MC520H { 5 DS, 10 US } [2 ch/US port]

The first line shows five downstream ports and 20 upstream ports. The second line shows 10 upstream ports at two channels per port. The third line contains the number of upstream ports and upstream channels per line card. In the fourth line, since there is no channelization of the DS, it is not mentioned when the "by modifier" is not used.

Usually having the attribute list buried in the port list reduces readability. NDL provides a syntax for extracting port level attributes out to the object list by using a per-port nomenclature. If there is only one port type, then "/port" is sufficient. Otherwise, the usage is "/port_type port". During calculations, attribute lists may be pushed into the port list, and then extracted out in the final expression.

Nested Channels

The following example shows how to express nested channelization such as DOCSIS logical channels within DOCSIS upstream QAM channels. Since NDL is focused more on connectivity, it is rare that such detail would be used, but it is available

MC520H { 5 x 20 } [4 logical ch/US ch]
MC520H { 5 x 10 } [4 logical ch/US ch, 2 ch/US port]

Using this approach, an infinite number of levels of channels can be defined.

Verbose Attribute Lists

Sometimes, details are of interest. The trick is to always retain readability so the end result is useful.

Let's take a CMTS line card that has frequency stacking in both directions, where logical channels exist in the upstream, and where certain channel parameters are of interest. The LC definition would be:

```
LC {
   5 DN [4 ch, 50 Mbps, 8 MHz, Annex A],
   10 UP [2 ch, 4 logical ch/ch, 27 Mbps, 6.4 MHz, 64QAM, ATDMA]
}
```

Many of these text attributes such as 64 QAM are not normally present since they cannot be used in an equation. However, they can be included if needed. Notice how NDL uses a list format for the port list to maintain readability. The opening curly bracket remains with the object name while the closing curly bracket is on its own line, aligned with the first character of the object's name. The port list is indented four spaces and has one port per line. The list is still separated by commas.

Redundancy

To achieve high availability (HA), a common practice is to include redundant line cards. The line card in use is often called the "working" card and the line card on standby is called the "protect" card. Redundancy groups are either 1+1 or N+1. The requirement for NDL is to indicate the presence of the line card, but not include the ports or attributes in any calculations.

Let's start with a non-redundant system. It is a CMTS with eight line cards.

```
CMTS { }
        = 8 * LC { 5 x 20 }
        = CMTS { 40 x 160 }
```

Now, to make the line cards redundant, the eight line cards now become a 7+1 group with seven working cards and one protect card. The quantity of the protect card(s) contains a "R" suffix to indicated redundancy and that the count should not be included in the calculations.

```
CMTS { }
        = (7 + 1 R) * LC { 5 x 20 }
        = CMTS { 35 x 140 }
```

The equation shows the intent of the system, and the result shows the correct number of ports.

Case Studies

These following case studies will introduce more nuances of NDL and show how more complicated network scenarios are handled as well as further formatting conventions.

Integrated CMTS

In an integrated CMTS (I-CMTS) deployment, all RF ports are contained within the CMTS chassis.

Say a basic I-CMTS consisted of eight cable line cards. That can be expressed in NDL as:

```
CMTS { }
        = 8 * Cable_LC { 5 x 20 }
        = CMTS { 40 x 160 }
```

This example assumes one channel per port. If two channels of frequency stacking are used on the upstream ports, the example becomes:

```
CMTS { }
        = 8 * Cable_LC { 5 x 10 } [2 ch/US port]
        = CMTS { 40 x 80 } [40 ch x 160 ch]
```

Notice how in the first line, the extracted port attribute keeps the port list more readable. Also notice in the second line how the NDL "by modifier" in the port list and attribute list to keep the presentation of the information simple.

Full CMTS

NDL can be used to describe the configuration of a system. This is usually more detail than is needed, but it can be useful for system documentation. This example will use the Cisco uBR10012 CMTS.

```
uBR10012_CMTS { }
        = 8 * MC520H { 5 x 20 }
        + 2 * DTI_LC { 2 DTI }
        + 2 * PRE { Serial }
        + 1 * SPA_Carrier { 2 SPA_I/F }
        + 2 * DEPI_SPA { SPA_I/F, GE [24 ch, DEPI] }
        + 2 * GE_LC { 1 GE } [WAN]
        = uBR10012_CMTS {
                40 x 160, 2 DTI, 2 GE [DEPI], 2 GE [WAN], Serial
          }
```

This CMTS has eight cable line cards, two DOCSIS Timing Interface (DTI) cards, two supervisor cards, one carrier card that carries two downstream external PHY interface (DEPI) shared port adapters (SPAs), and two Ethernet line cards. The SPA interfaces are connected and thus cancelled themselves out.

In the next example, a redundant system with a slightly different configuration is built. The RF redundancy is presumed to be internal, while the other ports have external connectivity for redundancy.

```
uBR10012_CMTS { }
        = (7 + 1 R) * LC { 5 x 20 }
        + (1 + 1 R) * DTI_LC { 2 DTI }
        + (1 + 1 R) * PRE { Serial }
        + (1 + 1 R) * SPA_Carrier { 2 SPA_Bay }
        + (1 + 1 R) * DEPI_SPA { SPA_I/F, GE [24ch, DEPI] }
        + (1 + 1 R) * GE_SPA { SPA_I/F, 1 GE } [WAN]
        = uBR10012_CMTS {
            35 x 140,
            (1 + 1 R) DTI,
            (1 + 1 R) GE [DEPI],
            (1 + 1 R) GE [WAN]
          }
```

Note the use of the list format once the final description became too long. The final ending curly bracket lines up with the first character in the object name.

Modular CMTS Core

A modular CMTS (M-CMTS) has its downstream QAM modulator extracted from the CMTS and placed into a separate device called an Edge QAM (EQAM). The remainder of the CMTS is called a modular CMTS core. The interface between the modular CMTS core and the EQAM is Ethernet. The protocol that runs over the Ethernet interface is DEPI (downstream external PHY interface).

Let's say that the CMTS chooses to implement the modular CMTS downstream port on a plug-in card called a SPA that has one GE port and supports 24 channels of DEPI. That line card would be described as:

```
DEPI_SPA { 1 GE } [24 ch, DEPI]
```

The text string "DEPI" serves as an attribute in the attribute list that describes the function of the GE port. The use of the term DEPI implies that the:

- DOCSIS DS MAC is present
- DOCSIS DS PHY is not present
- Upstream DOCSIS MAC and PHY are not present.

If two of these were added to the previous CMTS, the result would be:

```
M-CMTS_Core { }
        = 8 * MC520H { 5 x 20 }
        + 2 * DEPI_SPA { 1 GE } [24 ch x 0 ch, DEPI]
        = M-CMTS_Core {
                8 * 5 x 8 * 20,
                2 * 1 GE
          } [
                (8 * 5) + (2 * 24) ch x (8 * 20) ch
          ]
        = M-CMTS_Core { 40 x 160, 2 GE [DEPI] } [88 ch x 160 ch]
```

NDL defines and uses a listed sum-of-products format where the + sign is aligned with the = sign and the * sign is embedded within the line. NDL math is used in the object list, the port list, and the attribute list. When the calculation is written out, the port list and attribute list have used a list format to show separate calculations for readability.

The final result is a very terse description of a complicated machine that has a true representation of both the number of ports and the number of DS and US channels.

Edge QAM

This is an example of a network device whose outputs are entirely uni-directional. It contains only downstream QAM channels and no upstream paths. For this example, let's assume 12 ports and four channels per port:

EQAM { 12 [4 ch] x 0 }
EQAM { 12 x 0 } [48 ch]

EQAM { 12 DS [4 ch] }
EQAM { 12 DS } [48 ch]

All of these forms are equivalent. If the function of the network component was not understood from the object name, a further attribute can be added:

EQAM { 12 x 0 } [48 ch, QAM]

If the line card also has Ethernet ports, the representation would be:

EQAM { 12 DS, 2 GE } [48 DS ch, QAM DS]

Note the attribute list had to use a modifier to associate the card level attribute with a particular port type.

Modular CMTS System

This example shows the power of NDL to definitely, yet tersely, describe two separate systems and then combine them into one larger system.

A modular CMTS consists of a modular CMTS core plus an EQAM.

```
M-CMTS { }
        = M-CMTS_Core {
                40 DS , 160 US , 2 GE [24 ch, DEPI, A wire]
          }
        + EQAM {
                12 DS [4 ch], 2 GE [24 ch,  DEPI, A wire]
          }

        = M-CMTS {
                40 DS [1 ch], 12 DS [4 ch], 160 US [1 ch],
                (2 GE – 2 GE) [A wire]
          }

        = M-CMTS {
                40 DS [1 ch], 12 DS [4 ch], 160 US [1 ch]
          }
```

The first line describes the ports of the M-CMTS core. The second line describes the ports of an EQAM. The third line shows the calculations. The fourth line describes the resulting M-CMTS system.

This example introduces the wire attribute. The wire attribute indicates ports that are connected internally to the system and thus cancel out. The syntax is a unique text string followed by the unit called “wire”. If additional attributes need to be assigned to a wire, a connection object or a bundle object could be created with its own attribute list. These objects would connect to network objects with the wire attribute.

Note that there are two types of DS RF ports in the final system. There are:

- one-channel DS RF ports on the M-CMTS Core, and
- four-channel DS RF ports on the EQAM.

They are listed differently by their attribute in the port list.

Service Groups

A portion of the HFC Plant is defined as a Service Group (SG), and is connected to a CMTS or an EQAM. A SG is one or more downstream frequencies and zero or more upstream frequencies that are to be used on one or more Fiber Nodes (FN). A Fiber Node is an HFC Plant transmission element that interfaces between the fiber and the coax. The significance of a FN is that is represents a fixed number of Households passed (HHP).

A common SG in DOCSIS is shown in Figure 1 that shows a SG with one DS port and four US ports.

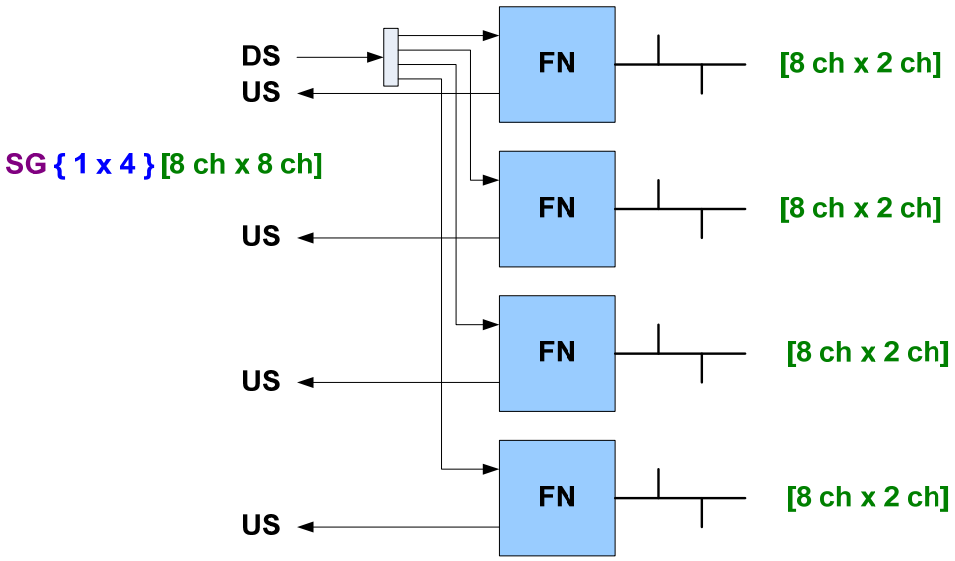

Figure 1: Service Group 1 x 4

The NDL representation is

SG { 1 x 4 }

If there is no explicit attribute list, the implication is that there is 1 ch/port on the DS and 1 ch/port on the US. Frequency allocations on a SG correspond to channels in an EQAM or CMTS, so NDL uses the channel notation to facilitate math.

What would the NDL representation be if the SG had eight frequencies on the DS and two frequencies on each upstream?

SG { 1 [8 ch] x 4 [2ch] }
= SG { 1 x 4 } [8 ch x 4 * 2 ch]
= SG { 1 x 4 } [8 ch x 8 ch]

SGs per CMTS, Case 1

How many SG will a CMTS support?

Using the basic CMTS from a previous example (with no redundancy):

CMTS { }
= 8 * LC { 5 x 20 }
= CMTS { 40 x 160 }

Lets take a simple definition of a SG of 1x4 with only one channel in the DS and US.

SG { 1 x 4 }

The SG per CMTS is solved by dividing the two objects.

SG/CMTS
= CMTS { 40 x 160 }
/ SG { 1 x 4 }
= MIN(40/1, 160/4)
= MIN(40, 40)
= 40 SG/CMTS

A minimum function is used as either the DS or US ports could become the limiting factor. Note that an object divided by an object is just a number with units. In this case, it is SG per CMTS. Although the math is simple in this example, NDL provides a completely documented set of calculations.

SGs per CMTS, Case 2

Service Group Definition:

Use the SG example from before that had multiple frequencies.

```
SG { }
        = SG { 1 [8 ch] x 4 [2 ch] }
        = SG { 1 x 4 } [8 ch x 8 ch]
```

CMTS Definition:

Let's use the M-CMTS Core definition from before that has the SPAs which add downstream MAC support.

```
M-CMTS_Core { }
        = 8 * LC { 5 x 20 }
        + 2 * SPA { 1 GE } [24 ch, DEPI]
        = M-CMTS_Core { 40 x 160 , 2 GE [DEPI] } [88 ch x 160 ch]
```

Note the WAN is not important in this calculation so it is not included.

Solution:

```
SG/CMTS
        = M-CMTS_Core { 40 x 160 , 2 GE [DEPI] } [88 ch x 160 ch]
        / SG { 1 x 4 } [8 ch x 8 ch]
        = MIN(
                40/1,           /* based upon DS ports */
                160/4,          /* based upon US ports */
                88/8,           /* based upon DS channels */
                160/8           /* based upon US channels */
          )
        = MIN( 40, 40, 11, 20 )
        = 11 SG/M-CMTS_Core         /* limited due to DS channels */
```

The equation divides the M-CMTS Core object by the SG object. This results in two port calculations and two channel calculations. Whichever is the critical resource will set the limit. In this case, the limitation was the number of DS channels.

Summary

NDL, the Network Design Language™, is a simple but powerful language for both representing network elements and for analyzing the connectivity and capacity of a network. All aspects of a network element can be described using an object, port or an attribute. Networks are defined using a syntax that breaks everything down to an object lists, port lists, and attribute lists.

Objects can be added together to create larger objects. Objects can also be divided by each other to calculate quantities. All math is basic algebra or spreadsheet type math.

NDL permits and even encourages shorthand. Only the minimum amount of information needs to be included as long as the result is clear. The formatting conventions of NDL are important as well to maximize readability and hence the detection of errors.

NDL Language Reference

NDL recognizes two usage cases:

- The first case is the informal, free-hand notation which applies to written calculations or whitepapers which might use NDL. In this case, certain elements of the language may be relaxed. In such cases, the text in the paragraphs contain content which may not need to be duplicated in an NDL equation. Further, it is usually more convenient to use white spaces in character strings rather than the underscore.

- The second case is a formal, machine-readable format where NDL is used as the input to a software application. In this case, NDL must be interpreted by a parser, so the language syntax needs to be precise and complete. There may also need to be some initialization code or prototyping code to set limits on configs that are to introduce any new object, port, or attribute types.

Object

Form:	Object { Port List [Attribute List] } [Attribute List]
Object	is a reserved name in NDL or a declared name
{ }	is a list of ports. Brackets are mandatory. Port listing is optional.
[ ]	is an optional list of attributes.
Formatting:	• Object names should have the first letter of each word in a name capitalized and should not contain any white spaces. Names based on abbreviations or acronyms should remain in their normal case. An underscore may be used between words in the name. • There must be one space between the object name and left curly bracket. • Line wraps should be avoided by using listing syntax. If a line wrap is necessary, a hanging indent should be used where subsequent line wraps are indented from the first line. Line wraps should be done after a comma, not in the middle of a list item.
Color:	• Object is in purple. RGB = 128,0,128
Informal Usage:	• The { } may be omitted if no ports are listed. • The object name may have spaces.
Examples:	• Ethernet_Switch { } • CMTS { } • uBR10012 { } • uBR10012_CMTS { 40 x 160, 2 DTI, 2 GE [DEPI], 2 GE [WAN] }

Port List

All objects have ports. Ports are the primary concern of NDL and have a reserved form. The port list is optional in informal NDL and mandatory in formal NDL, even if it is a null list. The port list only needs to include ports that are interesting to the current analysis. Excessive listing of irrelevant port types should be avoided.

Form:	{ count1 type1, count2 type2, … }
Count:	is the total port count for each port type for the object.
Type:	is a reserved or declared type of physical port. By default, ports are bidirectional and symmetrical.
By Modifier:	A special case is recognized for uni-directional ports. A double entry separated by the NDL "by modifier" (a lower case x) indicates a separate downstream and upstream port count (or egress and ingress port count for non-cable interfaces). For example, 8 indicates 8 bidirectional ports. 4 x 8 indicates 4 DS, 8 US ports for HFC Cable interfaces 4 x 8 indicates 4 egress and 8 ingress ports for non HFC interfaces. Egress is defined as away from the object. Ingress is defined as towards the object.
Formatting:	• There should be a space on the inside of each set of curly brackets. This allows individual ports to be more easily read. • The port list is a comma-separated list with a space or a combination of carriage return, tab, and spaces after the comma. • If empty curly brackets are used, they should contain one space "{ }".
Color:	• { port list } is in blue. RGB = 0, 0, 255
Informal Usage:	• Same as formal usage.
Examples:	• { 24 GE, 2 10GE } • { 2 GE, 5 x 20 }

Attribute List

Object and ports may have additional attributes of interest that need to be captured. Examples may include bandwidth values or port names. Attributes are separated by commas and are uniquely identified by the attribute units. Excessive listing of attributes is discouraged as it reduces readability. Generally, include only attributes that may be part of the equation, such as bandwidth.

Form:	[value1 units1, value2 units2, …]
Value:	The value of the attribute. If the attributes are different for the downstream and upstream directions, the NDL "by modifier" (lowercase x) notation may be used.
Units:	The units of the attribute. The value and units fields may be replaced by an attribute text string.
By Modifier:	The "by modifier" as defined for the port list can be used in the attribute list as well.
Port Modifier:	The port modifier allows port attributes to be included in the attribute list for the object. If the attribute applies to all ports in the port list, then the syntax is [Value Units/port]. Examples: [50 Mbps/port] [2 ch/port] If the attribute is for a particular port type, then the syntax is [Value Units/Type port]. Examples: [50 Mbps/DS port] [2 ch/US port] • If the object level attribute applies only to specific ports, then the modifier format is [Value Type Units] • If the attribute applies to a specific port type in the Port List, then the syntax is [Value Units/Type port].
Nested Attribute Modifier:	• Attributes may be nested. The classic case is channels within channels. The syntax then is [Value Units/Type Units]. Examples: [2 sub-ch/ch, 2 ch/port] [24 DS0/DS1, 28 DS1/DS3 port]
Usage:	• If the attribute list follows a port type, the attribute list applies to the port. (Port modifier is implied and not needed). • If the attribute list follows an object definition and there are no port modifiers, then the attributes apply to all ports on the object.
Informal Usage:	• Same as formal usage.

Formatting:	• There should be one space prior to the left square bracket and one space after the right square bracket when used inside the port list. There should be at least one space or tab prior to the left square bracket when used outside the port list. • There should be no space between the left square bracket and the first attribute, and no space between the last attribute and the right square bracket. • Empty square brackets "[ ]" should be omitted and not used.
Color:	• [Attribute List] is in green. RGB = 0, 128, 0
Examples:	• LC { 6 GE } [1 Gbps/port] • LC { 5 x 20 } [200 Mbps x 200 Mbps] • LC { 5 [40 Mbps] x 20 [10 Mbps] } • LC { 5 [6 MHz, 256QAM, Annex B, 38 Mbps] x 20 [3.2 MHz, 16QAM, 10 Mbps] } • LC { 5 x 10 } [2 ch/US port, 4 logical ch/US ch] • LC { 20 [3G x 1G] } • CMTS { 35 x 140 } ["San Jose CMTS3"] • CH { 3 GE, (2 GE – 2 GE) [A wire] } = CH { 3 GE }

Math Operators

NDL math can be used in the object list, the port list, or in the parameter list. Function calls are modeled after spreadsheet functions.

Form:	Object5 { } = 4 * Object1 { } + 3 * Object2 { } + 2 * (Object3 { } + Object4 { })
=	For equating an object to another set objects
+	For adding items
-	For subtracting items
*	For multiplying items by an integer
/	For dividing items by an integer or by one another
( )	For mathematical grouping of items
MIN( )	Minimum function. For choosing the between constraints.
MAX( )	Maximum function. For choosing the between constraints.
CEILING( )	Rounds up to an integer
FLOOR( )	Rounds down to an integer
// comment /* comment */	Single line comment. "\" continuation character allowed. Multi-line comment
Redundancy Modifier	• If the number of objects includes redundant objects, which are present but not in use, the quantity is modified to (N + M R) where N + M equals the original quantity, M represents the number of redundant objects, and R is a suffix denoting redundancy.
Formatting:	• One space or tab on either side of "=", "+", "*", "-", "/". • The "=" sign should be on the next line and indented one tab from the initial object declaration. • It is recommended to use a sum of products list format for readability. In the list format, the "+" sign is aligned vertically under the "=" sign, while the "*" sign is embedded within each line. • It is also recommended that a divide operation be written with the "/" beginning a new line.
Color:	• Black
Informal Usage:	• Same as formal usage.
Examples:	• (7 + 1R) * LC { 5 x 20 }

Prototypes

In formal NDL, there may be a need to declare an object prior to its use. By doing so, default attributes can be assigned to an object that don't need to be included in a main formula. This is done with prototyping. Prototyping can be used for defining object, ports, or attribute types.

Form:	Prototype Item Name Reference
Prototype	Use "Prototype" to start the syntax
Item	Use "Object", "Port", or "Unit"
Name	The intended name of the Item. When the Item is an Object, the Name takes the form Name { } and may include a full port list and attribute list.
Reference	The name of a previously defined or well-known Item from which all properties will be inherited unless explicitly overridden in the port list and/or attribute list. The reference is optional.
Port Count Modifier	Single value indicates the default value. (Min, Default, Max) indicates an operating range of values. Any of the three values may be omitted, but the commas must remain. For example (, , Max) would only specify a maximum value in the prototype.
Formatting:	• One line per Unit and Port prototype • One line per Object prototype if there is no port list or attribute list. Multiple lines otherwise with the { at the end of the first line and } on its own line, lined up with the first letter of the word prototype. • Prototypes do not contain math.
Color:	• Follow color standards for specific Objects, Ports, and Units. Everything else is in black.
Informal Usage:	• Prototyping is optional and usually not used for informal.

Examples:	Prototype Unit Mbps /* Megabits per second */
	
	Prototype Port DS_F /* Downstream Fiber */
	Prototype Port US_F /* Upstream Fiber */
	Prototype Port C /* Coax, bidirectional */
	
	Prototype Object FN {
	(1, 1, 1) DS_F /* DS Fiber ports */
	(1, 4, 4) US_F /* US Fiber ports */
	(1, 4, 4) C /* Coax ports */
	}
	
	Prototype Object uBR10012 { } CMTS { }
	
	Prototype Object_A { 10 GE } [1 Gbps/port, IOS_based]
	Prototype Object_B { 8 GE } Object_A { }

NDL Reserved Names

Object Names

General Objects

CH	Chassis
LC	Line card
Object	Generic. Mainly used for language reference
Network	Overall network
Router	Router
Switch	Ethernet switch.

HFC Cable Specific Objects

CM	Cable modem
CMTS	Cable modem termination system (DOCSIS)
EQAM	Edge QAM
FN	Fiber node
SG	Service group

Port Types

General Port Types

GE	Gigabit Ethernet
10GE	10 Gigabit Ethernet
DTI	DOCSIS timing interface
HSSI	High Speed Serial Interface
Serial	Default is RS-232 async
WAN	Wide area network

HFC Cable Specific Port Types

DS	HFC downstream port. Downstream ports are from PE (provider edge) to CE (customer edge) and may be ingress or egress, depending upon the object definition.
US	HFC upstream port. Upstream ports are from CE to PE and may be ingress or egress, depending upon the object definition. US attributes are generally independent of DS attributes.

Attribute Units

kbps	kilobits per second
Mbps	Megabits per second
Gbps	Gigabits per second
Tbps	Terabits per second
kHz	kilohertz
MHz	Megahertz
GHz	Gigahertz
HHP	Households passed
KHHP	1000 households passed
RS-232	Async Serial Interface
V.35	Sync Serial Interface
12in1	Cisco's Smart Serial Interface with 12 serial interfaces.
wire	used for connecting two or more ports together. Value is a text string. This triggers a subtraction of the tagged ports so that they do not propagate up to the higher level object.

Attribute Tags

DEPI	DEPI protocol from CableLabs
in	declares port to be an input-only port
out	declares port to be an output-only port
QAM	Quadrature Amplitude Modulation used.
WAN	Wide Area Network. Generic term for backhaul.
" "	Text string. Used for attaching a description to the port or object.

Notes

- NDL formulas are made up entirely of
  - Object Lists
  - Port Lists
  - Attribute Lists
- Object Lists are always mathematically listed.
- Port Lists and Attribute lists are comma-separated lists that can contain math.
- There are reserved names for the elements within each list. This is shown below.

Long Form:

Object { }
= *Quantity* * Object { *port_count port_type* } [*attribute_value attribute_units*]

Short Form:

Object { } = *Quantity* * Object { *Count Type* } [*Value Units*]

Acronyms and Abbreviations

ATDMA	Advanced time division multiple access
CE	Consumer edge
CM	Cable modem
CMTS	Cable modem termination system
DEPI	Downstream external PHY interface
DOCSIS	Data Over Cable Service Interface Specification
DTI	DOCSIS timing interface
DS	Downstream
EQAM	Edge QAM
FDM	Frequency division multiplexing
FN	Fiber node
FRU	Field replaceable unit
GE	Gigabit Ethernet
HA	High availability
HFC	Hybrid fiber coaxial
HHP	Households passed
I-CMTS	Integrated CMTS
LC	Line card
M-CMTS	Modular CMTS
MAC	Media access control
NDL	Network Design Language™
PE	Provider edge
PHY	Physical (layer)
QAM	Quadrature amplitude modulation
SFP	Small Form-factor Pluggable
SG	Service group
SPA	Shared port adapter
US	Upstream
WAN	Wide area network

About the Author

John T. Chapman is currently a Cisco Fellow and the Chief Architect for the Cable Business Unit at Cisco Systems in San Jose, California. As a founding member of the Cisco Cable BU, John has made significant contributions to Cisco and the cable industry through his pioneering work in DOCSIS and development of key technologies and concepts critical to the deployment of IP services over HFC plants. Included in these achievements are being the primary author of significant portions of the DOCSIS specifications These including DCC, UGS, UGS-AD, PHS for DOCSIS 1.1; originating and authoring DOCSIS Set-top Gateway (DSG); originating the downstream and upstream channel bonding concepts which evolved into DOCSIS 3.0; and originating DEPI and DTI for Modular CMTS. John has also published a number of ground breaking whitepapers on Multimedia Traffic Engineering (MMTE), DSG, QoS, high availability, M-CMTS, and Wideband DOCSIS, and is a respected and frequently requested speaker at industry events.

John graduated from the University of Alberta in Canada with a Bachelor in Science of Electrical Engineering in 1984. John began his formal career designing voice line, trunk, and ISDN cards for ROLM/IBM/Siemens and joined Cisco Systems in 1989. At Cisco, before joining the Cable BU in 1996, he was responsible for the development of HSSI (High-Speed Serial Interface) which is now an ANSI and ITU-T standard (V.12). Later projects included the industry's first ISDN BRI router, the design of the multi-protocol Smart Serial interfaces and HDLC ASICs used across the Cisco router product line, and co-architecting the VIP card in the Cisco 7500 product line.

John has over 50 patents issued or pending in a variety of technologies including telephony, VoIP, wide area networking, and broadband access for HFC cable networks. In his spare time, John enjoys spending time with his wife and two daughters. John is a 6th Degree Black Belt Master in Tae Kwon Do and enjoys white water canoeing and skiing.

Previous papers by John may be found at http://www.johntchapman.com.