VNF Requirement Updated Header Structure

Update of the structure for VNF Requirements to improve readability in areas which were nested.

Previously

1. Chapter titles were listed as the chapter number followed by the chapter title, surrounded by "**" at the start and finish, and followed by a line of "=====".
2. Sections would start with a lower case letter, followed by a line of "====".
3. Sub-sections would only be denoted by being surrounded with "**".
4. If a sub-section had any other sections listed under, it would be written by using tabs.

This method made it very difficult to actually follow the hierarchy of how the structure was laid out.

Newly Updated

1. Chapter titles will not need to start with the chapter number anymore.
2. Sections will not need lower case letters anymore.
3. Sub-sections will not need to be surrounded with "**" anymore.
4. If there are any sub-section levels included, you will not use tabs anymore.

The new structure will be using the RST file and a native Sphinx tool to create headings using the :numbered: argument in the toctree within the index file. Within the specific Chapter files there will be symbols under the titles of each header that will define how the structure will be set.

Important note: The line after the header text will be whatever symbol needed, **BUT** you must make sure that the length of the symbols will be the same length **OR** longer than the length of the text above it.

For example:

```
Service Design

------------------------------------
```

The newly updated structure will be as the following:

1. Chapters will be surrounded with "**" in the front and back **AND** Chapter headers will be followed by a line of '='.
    a. Sections will be followed by a line of '-'.
        i. Sub-sections will be followed by a line of '^'.
            1. Sub-sub-subsections will be followed by a line of '~'.
                a. Sub-sub-sub-subsections will be followed by a line of '+'.

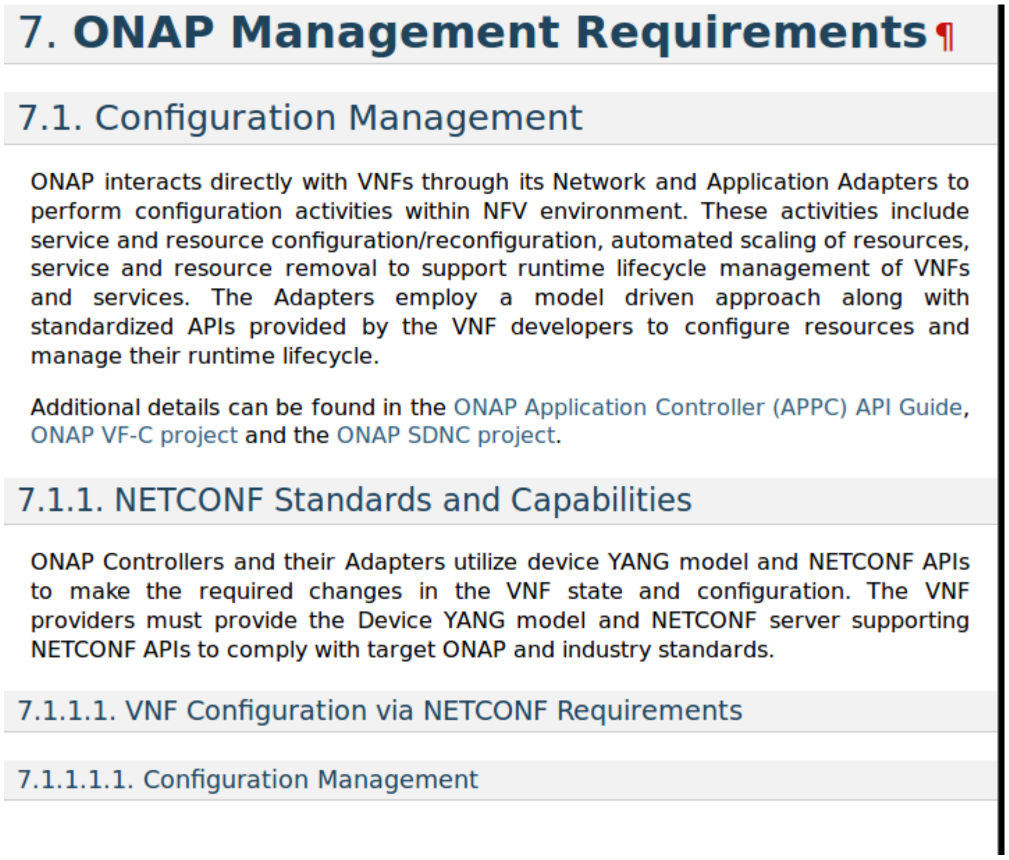
In simple terms the hierarchy is:

```
==========

----------------

^^^^^^^^^^^^^^^^

~~~~~~~~~~~

++++++++++
```

For Example:

```
**ONAP Management Requirements**
=================================

Configuration Management
---------------------------------------------------

ONAP interacts directly with VNFs through its Network and Application
Adapters to perform configuration activities within NFV environment.
These activities include service and resource
configuration/reconfiguration, automated scaling of resources, service
and resource removal to support runtime lifecycle management of VNFs and
services. The Adapters employ a model driven approach along with
standardized APIs provided by the VNF developers to configure resources
and manage their runtime lifecycle.

Additional details can be found in the `ONAP Application Controller (APPC) API Guide <http://onap.readthedocs.io
/en/latest/submodules/appc.git/docs/APPC%20API%20Guide/APPC%20API%20Guide.html>`_,
`ONAP VF-C project <http://onap.readthedocs.io/en/latest/submodules/vfc/nfvo/lcm.git/docs/index.html>`_ and the
`ONAP SDNC project <http://onap.readthedocs.io/en/latest/submodules/sdnc/northbound.git/docs/index.html>`_.

NETCONF Standards and Capabilities
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

ONAP Controllers and their Adapters utilize device YANG model and
NETCONF APIs to make the required changes in the VNF state and
configuration. The VNF providers must provide the Device YANG model and
NETCONF server supporting NETCONF APIs to comply with target ONAP and
industry standards.

VNF Configuration via NETCONF Requirements
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Configuration Management
+++++++++++++++++++++++++++
```

The RST above will generate:



7. **ONAP Management Requirements**¶

7.1. Configuration Management

ONAP interacts directly with VNFs through its Network and Application Adapters to perform configuration activities within NFV environment. These activities include service and resource configuration/reconfiguration, automated scaling of resources, service and resource removal to support runtime lifecycle management of VNFs and services. The Adapters employ a model driven approach along with standardized APIs provided by the VNF developers to configure resources and manage their runtime lifecycle.

Additional details can be found in the ONAP Application Controller (APPC) API Guide, ONAP VF-C project and the ONAP SDNC project.

7.1.1. NETCONF Standards and Capabilities

ONAP Controllers and their Adapters utilize device YANG model and NETCONF APIs to make the required changes in the VNF state and configuration. The VNF providers must provide the Device YANG model and NETCONF server supporting NETCONF APIs to comply with target ONAP and industry standards.

7.1.1.1. VNF Configuration via NETCONF Requirements

7.1.1.1.1. Configuration Management