Data Serialization

- Data serialization is the process of converting structured data to a standardized format that allows sharing or storage of the data in a form that allows recovery of its original structure
- It allows transfer of the data between different systems, applications and programming languages
- XML, JSON and YAML are human and machine readable, plain text data encoding formats

Data Serialization

- Data formats are mostly interchangeable
- Which one to use depends on support in the system it is being used with, and which is easiest for you

JSON JavaScript Object Notation

- First standardized in 2013.
- Easier for humans to read and work with than XML
- Can be imported directly into JavaScript, which is commonly used on the internet
- White space has no special meaning
- RESTful APIs often use JSON

JSON Data Types

- Object
- Array
- String
- Number
- Boolean
- Null

JSON Data Types: Object

FLACKBOX
www.flackbox.com

- An object is an unordered collection of key/value pairs.
- They are surrounded by curly braces {}.
- Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).
- Keys and values are separated by a colon.
- Each key/value pair is separated by a comma.

```
{
    “name”:”GigabitEthernet1”,
    “description”:”Internet link”,
    “enabled”:true
}
```

JSON Data Types: Array

- An array is an ordered list of values.
- They are surrounded by square brackets [].
- Values must be a valid JSON data type (string, number, object, array, boolean or null).

```
{
"name":"John",
"age":25,
“girlfriends":[ “Zoe", “Eve", “Amy" ]
}
```

JSON Nesting

```
{
  "address": [
    {
      "ip": "1.1.1.1",
      "netmask": "255.255.255.0"
    },
    {
      "ip": "10.255.255.1",
      "netmask": "255.255.255.0"
    }
  ]
}
```

JSON Data Types

FLACKBOX
www.flackbox.com

String:	`“name”:”GigabitEthernet1”`
Number:	`“Input Errors” : 3`
Boolean:	`“enabled” : true`
Null:	`“msec” : null`

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "GigabitEthernet1",
        "description": "Management",
        "enabled": true,
        "ietf-ip:ipv4": {
          "address": [
            {
              "ip": "10.10.20.48",
              "netmask": "255.255.255.0"
            }
          ]
        },
      },
      {
        "name": "GigabitEthernet2",
        "description": "Inside",
        "enabled": true,
        "ietf-ip:ipv4": {
          "address": [
            {
              "ip": "1.1.1.1",
              "netmask": "255.255.255.0"
            },
            {
              "ip": "10.255.255.1",
              "netmask": "255.255.255.0"
            }
          ]
        }
      }
    ]
  }
}
```

FLACKBOX
www.flackbox.com

XML eXtensible Markup Language

- Standardized in 1998.
- Widely used across the Internet.
- XML was designed to describe and transfer data, while HTML is focused on how to display data.
- White space has no special meaning in XML.
- <key>value</key> contained within object tags.

XML eXtensible Markup Language

FLACKBOX
www.flackbox.com

```
<?xml version =“1.0” encoding=“UTF-8” ?>
<interface xmlns=“ietf-interfaces”>
  <name>GigabitEthernet1</name>
  <description>Internet link</description>
  <enabled>true</enabled>
  <ipv4>
    <address>
      <ip>192.168.0.1</ip>
      <netmask>255.255.255.0</netmask>
    </address>
  </ipv4>
</interface>
```

YAML: YAML Aint Markup Language

- Often used in Python, Perl and Ansible
- Designed to be easily read by humans
- White space (indentation) is important
- Anything at a common indentation level is considered related at the same level
- Starts with ---
- key: value representation
- - indicates a list
- Ansible playbooks use YAML

YAML: YAML Aint Markup Language

```
---
ietf-interfaces:interface:
  name: GigabitEthernet1
  description: Internet link
  enabled: true
  ietf-if:ipv4:
    address:
    -  ip: 192.168.0.1
       netmask: 255.255.255.0
```