The NetLogger XML Data Service Interface Specification

Title:	NetLogger XML Data Service Interface Specification
Version:	1.3
Date:	December 1, 2023
Author:	K0JDD

Summary of Changes for Version 1.3

1. The use of HTTPS to access the API is now mandatory

2. The function GetPointer is removed. The pointer is returned as part of a call to GetCheckins

---

Summary of Changes for Version 1.2

1. The default **Interval** for GetPastNets has changed from 30 days to 7 days. If you need data past 7 days old you should combine the **Interval** parameter with the **NetNameLike** parameter to avoid a memory exhaustion error.

2. All API calls now return a **<ResponseCode>** as part of the XML structure, using standard HTTP/1.1 response codes as defined in RFC 2616. (https://datatracker.ietf.org/doc/html/rfc2616).   Table 1 on the final page lists the Response Codes returned by the API and the Element->Child or Element->Child->Sub Child where they can be found in the XML response.

3. An **Anti-Flooding** mechanism has been introduced on the server. This change was implemented due to a combination of DDOS attacks and poorly designed programs that use the API. When conditions warrant, the API will return both an error message and the response code **429 Too Many Requests.**

Most applications using the API are interrogating the server far too often.

As a result of how the official NetLogger Client program works, the data available through the API **does not change very frequently.** Calling GetActives Nets every 5-10 seconds wastes resources on your end as well as ours.

The following are our guidelines for interrogating the API.

API Routine	Recommended Calls / Minute
GetActiveNets	1
GetCheckins	3
~~GetPointer**~~	~~3~~
GetPastNets	1
GetPastNetCheckins	10

**As of Version 1.1, the <Pointer> is returned as part of GetCheckins. Using GetPointer is now deprecated and will be removed from the API in the future.

Summary of Changes for Version 1.1

1. Two new elements are added to the response to calls to GetCheckins.
   <CheckinCount> is the total number of <Checkin> entries on the list
   <Pointer> is the SerialNo of the currently working station

2. An optional query parameter has been added to GetActiveNets and GetPastNets: Specifying **&NetNameLike=string** will limit the response to only those nets whose name contains the specified string. The string should be URL encoded to avoid unexpected results.

Overview
This document describes the interface specification for access to the NetLogger XML Data Service. This service provides real-time access to certain information available from the NetLogger servers.

Access Methodology
The NetLogger XML service is implemented using standard HTTP protocols. Requests are made as query string arguments to a standard HTTP URL. Results are returned over HTTP as XML formatted data.

NetLogger XML Service Address
The NetLogger XML service is located at https://www.netlogger.org/api/.
The valid request scripts are:

GetActiveNets.php, GetCheckins.php, GetPastNets.php, and GetPastNetCheckins.php. GetPointer.php has been deprecated and will be removed in the future,

XML Data Structure
The NetLogger XML data structure follows the standard proposed by WWW.W3C.ORG. NetLoggers top-level node is the **<NetLoggerXML>** element. All responses from the NetLogger server are prefaced with this element. Four child node types are defined for this level. They are:

<Header>
<Error>
<ServerList>
<CheckinList>

Each of these nodes is described below. An important design pattern must be observed in order for your client program to maintain its highest level of reliability:

The XML data supplied by this interface may be extended at any time in a forward compatible manner. NetLogger may add new XML nodes and/or attributes at any time. Therefore, it is critically important that client programs carefully parse the data received in an "object=attribute" manner while ignoring any unknown nodes or attributes, and do so without raising an error condition. Your decoding strategy must make no assumptions regarding the number of nodes returned, the order in which nodes are returned, or the presence of nodes or attributes not previously defined.

The <Header> Node
The Header node contains information about the XML structure that is being returned including the Creation Date and Time, a Copyright Statement, the API Version, and the Time Zone that should be used to interpret all time and date values in the data.

The Header node ***may*** contain one or more **<Warning>** tags and these should be checked for and the contents of Warnings should be communicated in some fashion to the developer (not the requestor)

Sample Output:
```
<NetLoggerXML>
 <Header>
  <CreationDateUTC>Wed 01/06/2016 04:20:30</CreationDateUTC>
  <Copyright>The NetLogger System 2016</Copyright>
  <APIVersion>1.0</APIVersion>
  <TimeZone>UTC</TimeZone>
 </Header>
</NetLoggerXML>
```

The <Error> Node
If an invalid ServerName/NetName is submitted via an HTTP request, or a request is submitted via an HTTP request and the data set requested does not exist/is empty, the Error node will be returned along with a ResponseCode node.

Sample Output:
```
<NetLoggerXML>
 <Header>
  <CreationDateUTC>Wed 01/06/2016 04:21:20</CreationDateUTC>
  <Copyright>The NetLogger System 2016</Copyright>
  <APIVersion>1.0</APIVersion>
  <TimeZone>UTC</TimeZone>
 </Header>
 <Error>Not a Valid ServerName —> xxxx  or NetName —> xxxx</Error>     or   <Error> Query returned an empty result --> {query details}</Error>
<ResponseCode>404 Not Found</ResponseCode>
</NetLoggerXML>
```

The <ServerList> Node

The ServerList node is returned with the response to a request for a listing of all active or a range of past nets on all servers. It contains information about all of the NetLogger Servers, their internal names, and information about the active or past nets on each of these servers.

The ServerList node will contain a standard HTTP/1.1 Response Code and may have one or more children that are identified as **<Server>** nodes. These child Server nodes have one **<ServerName>** element, which identifies the internal name of the server, and zero or more children that are identified as **<Net>** Nodes.

The child Net nodes contain elements that describe all of the characteristics of a particular net including:

<NetName>	the original name of the net when opened
<AltNetName>	if the NCS changed the NetName, this is what it was changed to
<Frequency>	the radio frequency of the net
<Logger>	the station running the NetLogger program in Logging mode
<NetControl>	the station in charge of OTA net operations
<Date>	the date/time the net opened up on the NetLogger server
<Mode>	the operating mode

<Band>	the amateur band
<SubscriberCount>	the number of users currently monitoring the net
---and for past nets---	
<NetID>	A unique ID number for the net
<AIM>	Whether AIM was enabled or not (Y/N)
<UpdateInterval>	the AIM update interval in milliseconds
<srcIP>	the IP address of the computer/station that opened the net
<LastActivity>	the time of the last update or other significant activity for the net
<InactivitytTimer>	the time in minutes that the net must be inactive (no activity) before being auto-closed/assassinated
`<MiscNetParameters> (future use)`	
<ClosedAt>	The time the net was closed
<Assassinated>	whether or not the net was assassinated (Y/N)

Sample Output (Four active nets spread across two different servers – Past nets output is similar)

```
<NetLoggerXML>
 <Header>
  <CreationDateUTC>Wed 01/06/2016 03:33:54</CreationDateUTC>
  <Copyright>The NetLogger System 2016</Copyright>
  <APIVersion>1.0</APIVersion>
  <TimeZone>UTC</TimeZone>
 </Header>
 <ServerList>
 <ResponseCode>200 OK</ResponseCode>
  <Server>
   <ServerName>NETLOGGER</ServerName>
   <ServerActiveNetCount>3</ServerActiveNetCount>
   <Net>
    <NetName>Bryan Amateur Radio Club Net</NetName>
   <AltNetName>Bryan Amateur Radio Club Net</AltNetName>
    <Frequency>146.68</Frequency>
    <Logger>K5ZY - v2.4</Logger>
    <NetControl>K5ZY</NetControl>
    <Date>2016-01-06 01:51:22</Date>
    <Mode>FM</Mode>
    <Band>2m</Band>
    <SubscriberCount>13</SubscriberCount>
   </Net>
   <Net>
    <NetName>OMISS 80m SSB Net</NetName>
    <AltNetName>OMISS 80m SSB Net</AltNetName>
    <Frequency>3.824</Frequency>
    <Logger>N4JLT-KEN - v2.9.16W</Logger>
    <NetControl>N4JLT</NetControl>
    <Date>2016-01-06 01:25:27</Date>
    <Mode>SSB</Mode>
    <Band>80m</Band>
    <SubscriberCount>23</SubscriberCount>
   </Net>
   <Net>
    <NetName>Rag Chew Crew - Tailgaters</NetName>
    <AltNetName>Rag Chew Crew - Tailgaters</AltNetName>
    <Frequency>3.916 MHz</Frequency>
    <Logger>KE5GGY - v2.9.16W</Logger>
    <NetControl>KE5GGY</NetControl>
    <Date>2016-01-06 00:57:20</Date>
    <Mode>SSB</Mode>
    <Band>80m</Band>
    <SubscriberCount>7</SubscriberCount>
   </Net>
  </Server>
  <Server>
   <ServerName>NETLOGGER2</ServerName>
  <ServerActiveNetCount>1</ServerActiveNetCount>
   <Net>
    <NetName>omiss test</NetName>
    <Frequency>3.14159</Frequency>
    <Logger>K0JDD - v2.9.16M</Logger>
   <NetControl>K0JDD</NetControl>
   <Date>2016-01-06 03:02:19</Date>
   <Mode>SSB</Mode>
   <Band>80m</Band>
   <SubscriberCount>2</SubscriberCount>
  </Net>
 </Server>
 <Server>
  <ServerName>NetLogger3</ServerName>
 </Server>
 <Server>
  <ServerName>NetLogger4</ServerName>
 </Server>
 </ServerList>
</NetLoggerXML>
```

The <CheckinList> Node
The CheckinList node is returned in response to a request for a list of all information about a specific net on a specific server. It will also a standard HTTP/1.1 Response Code.

The <**ServerName>** and **<NetName>** submitted with the request are returned as data elements. There will also be zero or more **<Checkin>** child nodes which contain the following elements which correspond to the columns on the NetLogger screen:
<SerialNo>
<Callsign>
<State>
<Remarks>
<QSLInfo>
<CityCountry>
<FirstName>
<Status> (See the NetLogger client Help Files for an in depth discussion of the status field)
<County>
<Grid>
<Street>
<Zip>
<MemberID>
<Country>
<DXCC>
<PreferredName>

Sample Output
```
<NetLoggerXML>
 <Header>
  <CreationDateUTC>Wed 01/06/2016 04:13:20</CreationDateUTC>
  <Copyright>The NetLogger System 2016</Copyright>
  <APIVersion>1.0</APIVersion>
  <TimeZone>UTC</TimeZone>
 </Header>
 <CheckinList>
  < ServerName >NETLOGGER</ServerName>
  <NetName>omiss test</NetName>
  <ResponseCode>200 OK</ResponseCode>
  <CheckinCount>2</CheckinCount>
  <Pointer>1</Pointer>
  <Checkin>
   <SerialNo>1</SerialNo>
   <Callsign>K0JDD</Callsign>
   <State>MN</State>
   <Remarks>remarks</Remarks>
   <QSLInfo>GIB-9 V T N</QSLInfo>
   <CityCountry>Woodbury</CityCountry>
   <FirstName>JOHN</FirstName>
   <Status> </Status>
   <County>WASHINGTON</County>
   <Grid>EN34</Grid>
   <Street>4118 Woodlane Dr</Street>
  <Zip>55129</Zip>
  <MemberID>6066</MemberID>
  <Country>United States</Country>
  <DXCC>291</DXCC>
 <PreferredName>JD</PreferredName>
 </Checkin>
 <Checkin>
  <SerialNo>2</SerialNo>
  <Callsign>AC0ZG</Callsign>
  <State>CO</State>
  <Remarks>remarks also</Remarks>
  <QSLInfo>M LOTW-EQSL-Direct</QSLInfo>
  <CityCountry>Fort Collins</CityCountry>
  <FirstName>JOHN</FirstName>
  <Status> </Status>
  <County>Larimer</County>
  <Grid>DN70</Grid>
  <Street>4216 Picadilly Dr</Street>
  <Zip>80526</Zip>
  <MemberID>9548</MemberID>
  <Country>United States</Country>
  <DXCC>291</DXCC>
 <PreferredName/>
 </Checkin>
</CheckinList>
</NetLoggerXML>
```

Typical Usage – Active nets

There will typically be two requests required in order to retrieve data about active nets from the NetLogger System.

The first request will be made to the following url: (Case sensitive)
https://www.netlogger.org/api/GetActiveNets.php
This will return an XML structure containing the **<ServerList>** Node.

After Parsing the ServerList node to determine the names of any active nets of interest and the names of the server each active net resides on, a subsequent call will be made the following url: (Case sensitive)
http://www.netlogger.org/api/GetCheckins.php?ServerName=xxxxx&NetName=yyyy
where yyyy is the NetName of interest and xxxx is the ServerName that it resides on.

This will return an XML structure containing the **<CheckinList>** Node.

It is possible, and perhaps even desirable, to make one call to the GetActiveNets routine followed by multiple calls to the GetCheckins routine if you are interested in gathering data from multiple active nets at one time. *Be aware, however, that active nets can close at any time and that may result in a response with an empty **<CheckinList>** node.*

Typical Usage – Past Nets

There will typically be two requests required in order to retrieve data about past nets from the NetLogger System.

The first request will be made to the following url: (Case sensitive)
http://www.netlogger.org/api/GetPastNets.php?Interval=nn
This will return an XML structure containing the **<ServerList>** Node.
The optional parameter, Interval, specifies the number of days of past net activity to return.

After Parsing the ServerList node to determine the names and net ids of any past nets of interest and the names of the server each past net resides on, a subsequent call will be made the following url: (Case sensitive)
https://www.netlogger.org/api/GetPastNetCheckins.php?ServerName=xxxxx&NetName=yyyy&NetID=zzzz
where yyyy is the NetName of interest , zzzz is the corresponding NetID, and xxxx is the ServerName that it resides on.

This will return an XML structure containing the **<CheckinList>** Node.

It is possible, and perhaps even desirable, to make one call to the GetPastNets routine followed by multiple calls to the GetPastNetCheckins routine if you are interested in gathering data from multiple past nets at one time.

Table 1 - NetLogger API Response Codes

Parent	Child	Sub Child	Response Code		Notes
XMLDATA ->	ResponseCode		200	OK	
			400	Bad Request	Missing or Invalid Request Parameters
			401	Unauthorized	Requires Missing Permissions
			403	Forbidden	Method Not Allowed
			404	Not Found	Resource or Requested Data not found
			429	Too Many Requests	
			500	Database Error	
XMLDATA ->	ServerList ->	ResponseCode	200	OK	
XMLDATA ->	CheckinList ->	ResponseCode	200	OK	
			404	Not Found	Indicates an Empty Result
XMLDATA ->	MonitorList -> (See Appendix A)	ResponseCode	200	OK	
			404	Not Found	Indicates an Empty Result
XMLDATA ->	AIMTranscript -> (See Appendix A)	ResponseCode	200	OK	
			404	Not Found	Indicates an Empty Result
XMLDATA ->	Session -> (See Appendix B)	ResponseCode	200	OK	