TechNote

anynode®
The Software SBC

# anynode Route Supervision

October 15, 2019

## Contents

## Introduction

anynode implements a **RESTful** client to connect to an external WEB server to implement route supervision and / or send out notifications about call establishment or disconnect. Information are exchanged via **HTTP** or **HTTPS**. The content is **JSON** formatted.

## Authentication

For authentication HTTP basic and digest authentication are supported. Credentials must be configured in the anynode Frontend. The authentication is controlled by the WEB-Server. If TCP is used the initial HTTP request will not contain any authorization data. The WEB-Server may respond with a 401 and request basic or digest authentication. If TLS is used and a username and password are specified in anynode Frontend the initial HTTP request will contain the **'Authorization'** header for basic authentication. The WEB-Server may accept the authentication or respond with a 401 to request digest authentication.

## Routing Information Request

The following information is sent to request routing information from a WEB server:

- Source-Address (dial string and display name)
- Destination-Address (dial string and display name)
- Unique Identifier (created by anynode)

The WEB Servers response contains information if the call should be routed and optionally may modify the addresses. Available parameters are:

- Continue Route
- Ignore Route
- Reject Route
- Reason for reject
- Source-Address (optional dial string and / or display name)
- Destination-Address (optional dial string and / or display name)

If the WEB-Server processes the request a 200 OK response is expected. The JSON content defines how to route the call. Any other response (except the 401 request for authorization to the initial request) results in aborting or ignoring the route, depending on the configuration.

### 3.1 HTTP Method and URL

The HTTP method may be set to **GET**, **POST** or **PUT**. The default is GET. The URL suffix for routing information requests can be configured, the default is **"/anynode/route"**.

### 3.2 Accept (Continue) Route

To accept a route without any changes to source or destination address information the **"Continue Route"** flag must be set to **'true'** in the JSON content.

### 3.3 Modify Route

To accept and modify a route the "Continue Route" flag must be set to 'true' in the JSON content as well as the address information that should be changed. Address information that are not included in the JSON-Content or set to type 'null' will not be modified. To delete an existing address information (e.g. a display name) the parameter must be set as empty string.

### 3.4 Ignore Route

To ignore a route the "Ignore Route" flag must be set in the JSON content. The "Continue Route" flag must be either set to 'false' or must not be included. Depending on the routing configuration in anynode the next matching route is processed, or the call is disconnected.

### 3.5 Reject Route

To reject a route the "Reject Route" flag must be set in the JSON content. The "Continue Route" and the "Route Ignore" flags must be either set to 'false' or must not be included. Optionally the reason for the reject can be specified. The following reasons are available:

- Success
- Wrong Dial String / User not found etc.
- No Permission
- Congestion, all lines used
- Device (Network) error
- Busy
- Redirected
- Not Responding (No Answer)
- Not Selected
- Rejected by user
- Terminated by user
- Media Negotiation Error
- Other (unspecified Error)
- Domain Specific (20 codes)

## Notifications

Notifications can be sent to inform the WEB server, that a call is initiated, connected or disconnected. For notifications anynode does not expect a response. Any received response is ignored.

### 4.1 HTTP Method and URL

The HTTP method may be set to GET, POST or PUT. The default is POST. The URL suffix for notifications can be configured, the default is "/anynode/notification".

### 4.2 Call Assignment

The assignment between routing information requests and notifications is done via a unique identifier created by anynode. The identifier is contained in the JSON content of any request sent by anynode to the WEB-Server.

### 4.3 Call Initiated

The format of the "Call Initiated Notification" is the same as a routing information request. The difference is that anynode does not expect a response and directly continues with the routing as configured.

### 4.4 Call State

The JSON Response for call state contains the flags **"active"** and **"terminated"**. They will be set once and never reset. The notification is sent when the call state changes to **connected ("active": true)** and **disconnected ("terminated": true)**. The following combination of the flags may occur.

| "active" | "terminated" | Description |
|---|---|---|
| False | False | Will not occur. |
| True | False | Call is connected |
| False | True | Call failed |
| True | True | Call was connected and is now disconnected |

When the "terminated" flag is set to 'true', the parameter "reason" specifies why the call is disconnected.

- Success (Call was connected and is now disconnected)
- Busy
- Not Responding (No Answer)
- Rejected
- Error (Other Reason)

# JSON Schemas

All information is exchanged as **HTTP content** in **JSON-Format**.
There are Schemas for:

- Routing information request and notify call initiated
- Routing information response
- Notify call state

There are default schemas available and it is recommended to use them. If the WEB Server requires different JSON content a customized schema can be set in the configuration. If different keywords for the parameter are also required, they can be configured as well. The configuration is currently done via so called “unbound settings”.

## 5.1 Default Schemas

This chapter specifies the default schemas for routing information and call state notification. Note that the JSON content sent by the Web server as routing information response may include additional keywords and parameter. Any keywords unknown to anynode will be silently discarded.

### 5.1.1 Routing Information Request

The default schema for the routing information request is:

```
{
  "type":"object",
  "properties":{
    "identifier":{"type":"string"},
    "sourceAddress":{
      "type":"object",
      "properties":{
        "dialString":{"type":"string"},
        "displayName":{"type":"string"}
      }
    },
    "destinationAddress":{
      "type":"object",
      "properties":{
        "dialString":{"type":"string"},
        "displayName":{"type":"string"}
      }
    }
  }
}
```

### 5.1.2 Routing Information Response

The default schema for the response to a routing information request is:

```
{
  "type": "object",
  "properties": {
    "routeContinue": { "type": "boolean" },
    "routeIgnore":   { "type": "boolean" },
    "routeReject":   { "type": "boolean" },
    "rejectReason":  {
      "type": "string",
      "enum": [
        "success",
        "dialString",
        "networkPermission",
        "networkCongestion",
        "networkEquipment",
        "busy",
        "redirected",
        "notResponding",
        "notSelected",
        "rejected",
        "userTerminated",
        "mediaNegotiation",
        "error",
        "domainSpecific0",  "domainSpecific1",  "domainSpecific2",
        "domainSpecific3",  "domainSpecific4",  "domainSpecific5",
        "domainSpecific6",  "domainSpecific7",  "domainSpecific8",
        "domainSpecific9",  "domainSpecific10", "domainSpecific11",
        "domainSpecific12", "domainSpecific13", "domainSpecific14",
        "domainSpecific15", "domainSpecific16", "domainSpecific17",
        "domainSpecific18", "domainSpecific19"
      ]
    },
    "sourceAddress": {
      "type": "object",
      "properties": {
        "dialString":  { "type": "string" },
        "displayName": { "type": "string" }
      }
    },
    "destinationAddress": {
      "type": "object",
      "properties": {
        "dialString":  { "type": "string" },
        "displayName": { "type": "string" }
      }
    }
  }
}
```

### 5.1.3 Call State Notification

```
{
  "type": "object",
  "properties": {
    "identifier": { "type": "string" },
    "active": {"type": "boolean" },
    "terminated": {"type": "boolean" },
    "reason": {
      "type": "string",
      "enum": [
        "success",
        "busy",
        "notResponding",
        "rejected",
        "error"
      ]
    }
  }
}
```

## 5.2 Custom Schemas

Using a custom schema requires to specify the assignment between JSON keywords and parameters interpreted by anynode. This chapter lists the keywords for each schema and the configuration parameter names belonging to the keyword. At the end of the chapter a sample custom schema with the configuration parameter is shown. Parameter may occur in nested objects. The name to identify a keyword uses a dotted notation in the format **<object name>.<keyword>** where **<object name>** may occur multiple times.

### 5.2.1 Routing Information Request

The configuration parameter to specify a schema for the routing information request is **"jsonRequestSchema"**. The table below specifies the configuration parameter names and the keywords used for the default schema.

| Configuration parameter name | Value for default schema |
|---|---|
| jsonKeyIdentifier | identifier |
| jsonRequestKeySourceDialString | sourceAddress.dialString |
| jsonRequestKeySourceDisplayName | sourceAddress.displayName |
| jsonRequestKeyDestinationDialString | destinationAddress.dialString |
| jsonRequestKeyDestinationDisplayName | destinationAddress.displayName |

### 5.2.2 Routing Information Response

The configuration parameter to specify a schema for the routing information response is **"jsonResponseSchema"**. The table below specifies the configuration parameter names and the keywords used for the default schema.

| Configuration parameter name | Value for default schema |
|---|---|
| jsonResponseKeyRouteContinue | routeContinue |
| jsonResponseKeyRouteIgnore | routeIgnore |
| jsonResponseKeyRouteReject | routeReject |
| jsonResponseKeyRejectReason | rejectReason |
| jsonResponseKeySourceDialString | sourceAddress.dialString |
| jsonResponseKeySourceDisplayName | sourceAddress.displayName |
| jsonResponseKeyDestinationDialString | destinationAddress.dialString |
| jsonResponseKeyDestinationDisplayName | destinationAddress.displayName |
| jsonEnumReasonSuccess | success |
| jsonEnumReasonDialString | dialString |
| jsonEnumReasonNetworkPermission | networkPermission |
| jsonEnumReasonNetworkCongestion | networkCongestion |
| jsonEnumReasonNetworkEquipment | networkEquipment |
| jsonEnumReasonBusy | busy |
| jsonEnumReasonRedirected | redirected |
| jsonEnumReasonNotResponding | notResponding |
| jsonEnumReasonNotSelected | notSelected |
| jsonEnumReasonReject | rejected |
| jsonEnumReasonUserTerminated | userTerminated |
| jsonEnumReasonMediaNegotiation | mediaNegotiation |
| jsonEnumReasonError | error |
| jsonEnumReasonDomainSpecific | domainSpecific |

It is expected that to the keyword of the parameter **"jsonEnumReasonDomainSpecific"** an index between 0 and 19 is appended.

### 5.2.3 Call State Notification

The configuration parameter to specify a schema for the call state notifications **"jsonNotifySchema"**. The table below specifies the configuration parameter names and the keywords used for the default schema.

| Configuration parameter name | Value for default schema |
|---|---|
| jsonKeyIdentifier | identifier |
| jsonNotifyKeyCallActive | active |
| jsonNotifyKeyCallTerminated | terminated |
| jsonNotifyKeyReason | reason |
| jsonEnumReasonSuccess | success |
| jsonEnumReasonBusy | busy |
| jsonEnumReasonNotResponding | notResponding |
| jsonEnumReasonReject | rejected |
| jsonEnumReasonError | error |

### 5.2.4 Sample Routing Information Request

The following is a sample how to overwrite the schema and set the keyword accordingly.

#### 5.2.4.1 Schema

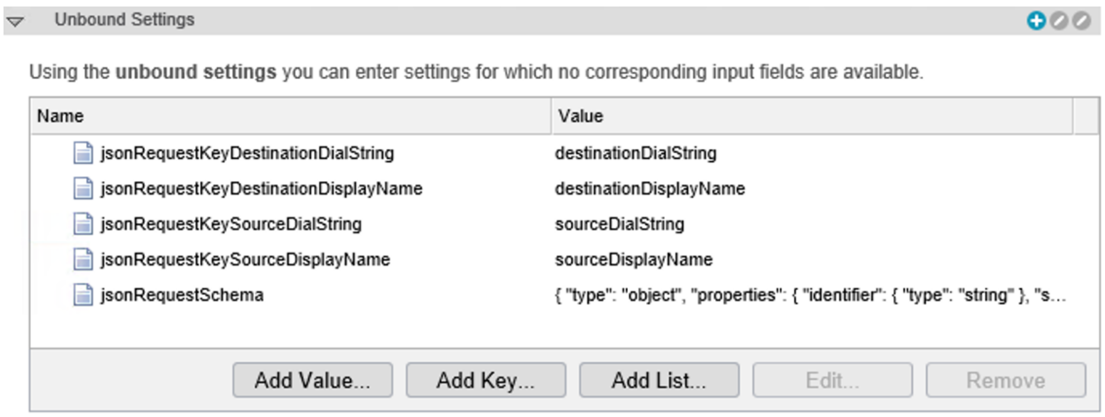
```
{
  "type": "object",
  "properties": {
    "identifier": { "type": "string" },
    "sourceDialString":  { "type": "string" },
    "sourceDisplayName": { "type": "string" },
    "destinationDialString":  { "type": "string" },
    "destinationDisplayName": { "type": "string" }
  }
}
```

#### 5.2.4.2 Keyword Assignment

| Configuration parameter name | Value for schema above |
|---|---|
| jsonKeyIdentifier | identifier |
| jsonRequestKeySourceDialString | sourceDialString |
| jsonRequestKeySourceDisplayName | sourceDisplayName |
| jsonRequestKeyDestinationDialString | destinationDialString |
| jsonRequestKeyDestinationDisplayName | destinationDisplayName |

#### 5.2.4.3 Configuration

The parameters are configured in the **"Unbound Settings"** of a **"REST Client Route Supervision"**. The screenshot below shows the sample configuration.



Using the **unbound settings** you can enter settings for which no corresponding input fields are available.

| Name | Value |
|---|---|
| jsonRequestKeyDestinationDialString | destinationDialString |
| jsonRequestKeyDestinationDisplayName | destinationDisplayName |
| jsonRequestKeySourceDialString | sourceDialString |
| jsonRequestKeySourceDisplayName | sourceDisplayName |
| jsonRequestSchema | { "type": "object", "properties": { "identifier": { "type": "string" }, "s... |

# Exclusion of Liability

### Third Party Disclaimer and Limitations

„Web Toolkit“, developed by Google (http://code.google.com/webtoolkit/).
„Smart GWT“, developed by Isomorphic Software, Inc. (http://www.smartclient.com/).
„Jetty“, developed by Mort Bay Consulting Pty Ltd (http://mortbay.com/).
„Java Native Access“, developed at github.com (https://github.com/twall/jna).
„Apache Commons IO“, developed by the Apache Software Foundation (http://www.apache.org/).
„Guava Libraries“, developed by Google (http://code.google.com/p/guava-libraries/).
„LDAP SDK“, developed by Unbound ID (https://www.unboundid.com/products/ldap-sdk/).
„Freemarker“, developed at freemarker.org (http://freemarker.org/).
„jsoup“, developed by Jonathan Hedley (http://jsoup.org/).
„OpenSSL“, developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/), written by Eric Young (eay@cryptsoft.com) and written by Tim Hudson (tjh@cryptsoft.com). [Windows only]
„Opus codec“, developed by the Xiph Foundation (http://www.opus-codec.org/license/).
„SQLite“, developed at sqlite.org (https://sqlite.org/).
„Java Runtime“, developed by Oracle Corporation (JRE License Terms). [Windows only]
„SILK codec“, developed by Skype Limited (https://www.skype.com/)

### anynode-Frontend

This product includes software developed by Google (http://code.google.com/webtoolkit/)
This product includes software developed by Isomorphic Software, Inc. (http://www.smartclient.com/)
This product includes software developed by Mort Bay Consulting Pty Ltd (http://mortbay.com/)
This product includes software (JNA) developed at github.com (https://github.com/twall/jna)
This product includes software developed by the Apache Software Foundation (http://www.apache.org/)