NAST

Documentation

Doc. version:	2.0.1	**Doc. status:**	Final
Doc. date:	01.12.2015	**Doc. name:**	NAST-Documentation-DNS Services-V2.0.1-2015-12-01

Imprint

Authors	Department	Phone	E-mail
DNS Services	DNS Services	+49-69-27 235 272	info@denic.de

Document Release

Document version	Released by	Released on
2.0.1	DNS Services	01.12.2015

Distribution List

Name
Members
Public

Contents

1. General Information

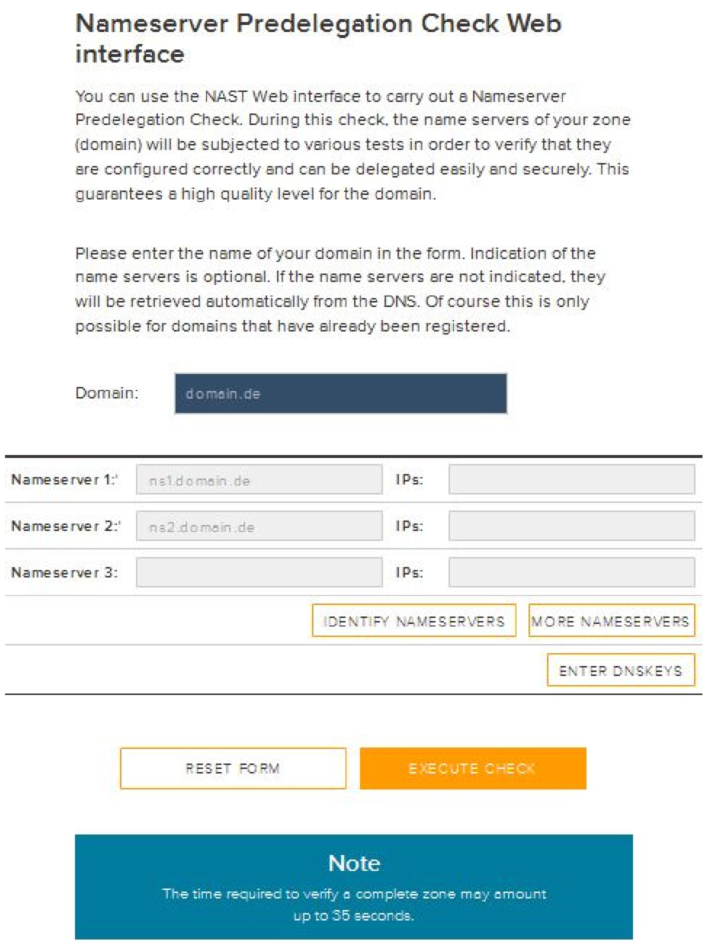
1.1 About this Document

This document describes the software (source code, REST service and command-line interface) and the web client DENIC makes available for checking name servers with regard to the correctness of delegation data for second level domains under .de and 9.4.e164.arpa.

Important information is displayed separately and highlighted.

Vertical red lines in the margins draw attention to new passages.

The NAST web client (NAST = NAme Server Tester) can be accessed at *http://nast.denic.de*



Nameserver Predelegation Check Web interface

You can use the NAST Web interface to carry out a Nameserver Predelegation Check. During this check, the name servers of your zone (domain) will be subjected to various tests in order to verify that they are configured correctly and can be delegated easily and securely. This guarantees a high quality level for the domain.

Please enter the name of your domain in the form. Indication of the name servers is optional. If the name servers are not indicated, they will be retrieved automatically from the DNS. Of course this is only possible for domains that have already been registered.

Domain:	domain.de		
Nameserver 1:'	ns1.domain.de	IPs:	
Nameserver 2:'	ns2.domain.de	IPs:	
Nameserver 3:		IPs:	

Note
The time required to verify a complete zone may amount up to 35 seconds.

1.2 Motivation

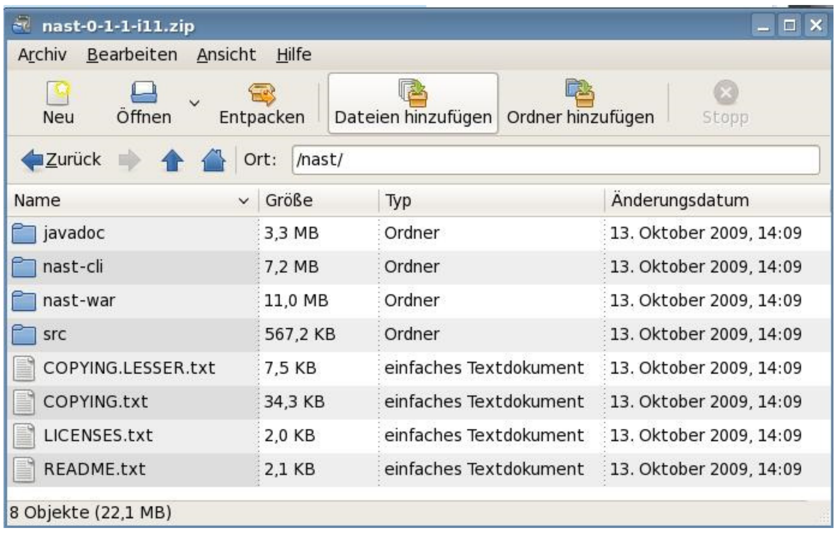
The software package NAST (NAme Server Tester) enables you to carry out the tests DENIC executes within the scope of the Nameserver Predelegation Check yourself in order to avoid initial setup errors and to facilitate the delegation of second level domains under .de and 9.4.e164.arpa.

For the requirements that must be met, refer to the documentation DENIC-23 – Nameserver Predelgation Check at *http://nast.denic.de*

DENIC makes available a software package at *http://nast.denic.de*.

2. Contents of the Software Package

The ZIP archive of NAST comprises the following:



Name	Größe	Typ	Änderungsdatum
javadoc	3,3 MB	Ordner	13. Oktober 2009, 14:09
nast-cli	7,2 MB	Ordner	13. Oktober 2009, 14:09
nast-war	11,0 MB	Ordner	13. Oktober 2009, 14:09
src	567,2 KB	Ordner	13. Oktober 2009, 14:09
COPYING.LESSER.txt	7,5 KB	einfaches Textdokument	13. Oktober 2009, 14:09
COPYING.txt	34,3 KB	einfaches Textdokument	13. Oktober 2009, 14:09
LICENSES.txt	2,0 KB	einfaches Textdokument	13. Oktober 2009, 14:09
README.txt	2,1 KB	einfaches Textdokument	13. Oktober 2009, 14:09

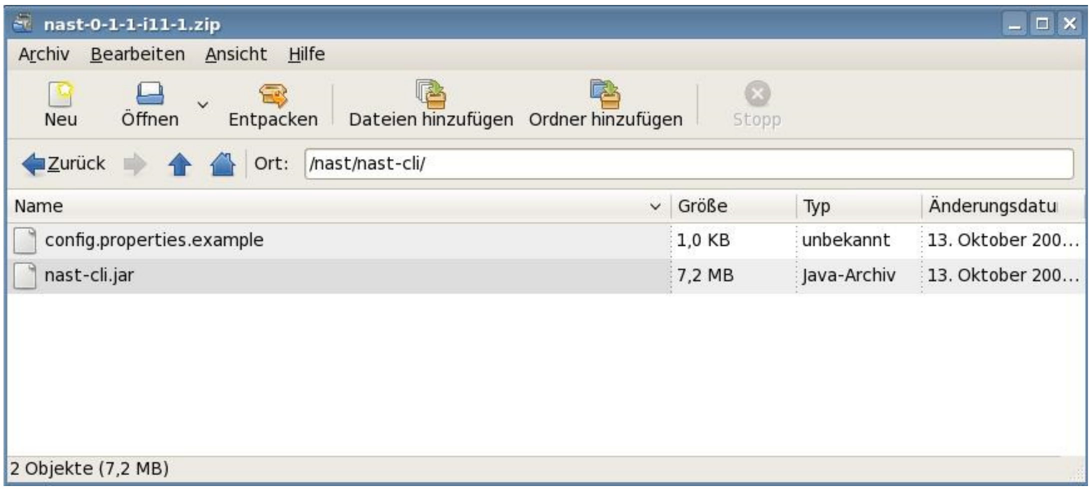
8 Objekte (22,1 MB)

CHANGELOG.txt	Changelog (Log of changes)
COPYING.LESSER.txt	Contains the GNU Lesser General Public License under which DENIC provides the software
COPYING.txt	GNU General Public License
LICENSES.txt	Lists other 3rd party libraries that are used
nast-war	The REST service to be included in a Servlet engine
javadoc	API documentation for NAST in HTML format
src	Program source code of NAST that is provided under the LGPL[1] (GNU Lesser General Public License) version 3.0.
nast-cli	Command-line interface and optional configuration

[1] http://www.gnu.org/licenses/

3. Command-Line Interface

You will find the command-line interface of NAST in the *nast-cli* directory:


nast-0-1-1-i11-1.zip
Archiv Bearbeiten Ansicht Hilfe
Neu Öffnen Entpacken Dateien hinzufügen Ordner hinzufügen Stopp
Zurück
Ort: /nast/nast-cli/

Name	Größe	Typ	Änderungsdatu
config.properties.example	1,0 KB	unbekannt	13. Oktober 200…
nast-cli.jar	7,2 MB	Java-Archiv	13. Oktober 200…

2 Objekte (7,2 MB)

File	Meaning
nast-cli.jar	Standalone Application
config.properties.example	Configuration file (optional)

3.1 System Requirements

The system on which NAST shall be installed must satisfy the following requirements:

- Java Runtime Environment, version 1.7 or higher,
- access to a default resolver,
- IPv4 or IPv6 access to the DNS.

3.2 Installation

The command-line interface can be called directly via the command line with the command `java -jar nast-cli.jar <Option>`.

3.3 Configuration

In the `config.properties` file you can configure the following technical parameters:

Parameter	**Meaning**	**Value Range**	**Mandatory / Optional**
default.resolver.interface	Host name of default resolver *(Default: localhost)*	"localhost", name of interface or IP address	optional
default.resolver.port	Port number of default resolver *(Default: 53)*	0 – 65535	optional
global.unicast.ipv6.url	URL for loading the "IPv6 Global Unicast Address Assignments" *(Default: http://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xml*) Please note that the URL can also be of type file://<file_name>	Valid URL in accordance with RFC3986	mandatory

If you do not configure a default resolver, NAST will try to automatically determine the default resolver of the platform. The application will follow the procedure documented at:

http://www.dnsjava.org/dnsjava-current/doc/org/xbill/DNS/ResolverConfig.html

3.4 Options

The next paragraphs give a survey of the options available for the command-line interface.

3.4.1 Domain

With `-d <domain>` the domain to be checked is handed over.

You can use this option only in combination with the option `-n <Nameserver>`.

3.4.2 Name Server

With the option `-n` the name servers to be checked are handed over.
The host name and the IPv4 and/or IPv6 address will be separated by a blank.
The various host names will be separated by commas ",".

Example: Parameter for –n

```
java -jar nast-cli.jar -d denic.net -n "ns1.denic.de,ns2.denic.de"

java -jar nast-cli.jar -d denic.de -n "ns1.denic.de 81.91.170.1
2001:608:6:6:0:0:0:11,ns5.denic.net"

java -jar nast-cli.jar -d denic.de -n "ns1.denic.de 81.91.170.1
2001:608:6:6:0:0:0:11,ns2.denic.de 193.171.255.36,ns3.denic.de
87.233.175.19,ns4.denic.net,ns5.denic.net"
```

3.4.3 Resolver

With `-i <Resolver>` you can include an optional resolver deviating from the default/configured resolver for the query:

Example: Optional resolver

```
java -jar nast-cli.jar -d denic.net -n "ns1.denic.de,ns2.denic.de" –i resolver.denic.de –p
9053
```

Parameter `-p <Port>` is mandatory.

3.4.4 Port

With `-p <Port>` you can state a port deviating from the default/configured port 53 for your query:

Examle: Optional resolver port

```
java -jar nast-cli.jar -d denic.net -n "ns1.denic.de,ns2.denic.de" –i resolver.denic.de –p
9053
java -jar nast-cli.jar -d denic.net -n "ns1.denic.de,ns2.denic.de" –p 9053
```

3.4.5 Optional format of output

With `-f XML` or `-f JSON` you can determine the output format.

If you choose `-f XML` output will be in the XML format:

Example: Output in XML-Format.

```
[lorelai@starshollow]$ java -jar nast-cli.jar -d denic.net -n "ns1.denic.de,ns2.denic.de" –f
XML
Version: 1-0--1-i18 (2010-01-08 12:43)
Applying JVM/OS stub as default recursive resolver
<predelegation-check success="true" xmlns="http://schema.denic.de/rng/nast"/>
```

If you choose `-f JSON` output will be in the JSON format:

Example: Output in JSON-Format.

```
[lorelai@starshollow]$ java -jar nast-cli.jar -d denic.net -n "ns1.denic.de,ns2.denic.de" –f
JSON
Version Version: 1-0--1-i18 (2010-01-08 12:43)
Applying JVM/OS stub as default recursive resolver
{"success":true}
```

3.4.6 Help

With `-h` you get access to general help texts.

Beispiel: Ausgabe im XML-Format.

```
[lorelai@starshollow]$ java -jar nast-cli.jar -d denic.net -n "ns1.denic.de,ns2.denic.de" –f
JSON
No config file "config.properties" found. Use example "config.properties.example" file to
build one if applicable.
EXAMPLE: java -jar nast.jar –h -d DOMAIN -f FORMAT -h -n NAMESERVER -ri RESOLVER
-rp PORT -v
 -d DOMAIN     : Name of domain to execute tests against (like "test.xx")
 -f FORMAT     : Optional format of output ("XML" or "JSON")
 -h            : Print this command line parameter help
 -n NAMESERVER : Comma-separated list of nameserver parameters (like "ns1.test.x
                x 192.168.1.1, ns2.other.xx")
 -i RESOLVER   : Overrides configured interface of default resolver (like "192.1
                68.1.1" or "myresolver.xx")
 -p PORT       : Overrides configured port of default resolver (like "53")
 -v            : Be verbose
```

3.4.7 DNSSEC-options

If `–D DNSSEC` is stated, the checks will include DNSSEC-specific Predelegation Checks.
If `–D` is not stated, the default is assumed and no DNSSEC-specific checks will be executed.

Beispiel: Predelegation-check with DNSSEC

```
java -jar nast-cli.jar -d denic.net -n "ns1.denic.de,ns2.denic.de" –D DNSSEC -k "257 3 5
AwEAAdDECajH
aTjfSoNTY58WcBah1BxPKVIHBz4lfLjfqMvium4lgKtKZLe97DgJ5/NQrNEGGQmr6fKvUj67
cfrZUojZ2cGRizVhgkOqZ9scaTVXNuXLM5Tw7VWOVIceeXAuuH2mPliEV6MhJYUsW6d
vmNsJ4XwCgNgroAmXhoMEiWEjBB+wjYZQ5GtZHBFKVXACSWTiCtddHcueOeSVPi5W
H94VIubhHfiytNPZLrObhUCHT6k0tNE6phLoHnXWU+6vpsYpz6GhMw/R9BFxW5PdPFI
WBgoWk2/XFVRSKG9Lr61b2z1R126xeUwvw46RVy3hanV3vNO7LM5HniqaYclBbhk="
```

4. NAST Web Service

The NAST web service is a web application running in a Servlet container and implemented as a RESTful web service. REST (REpresentational State Transfer) is an architectural style that uses the semantics and methods of the HTTP protocol for the communication between client and server. You will find an introduction to REST at Wikipedia[2], for example. You do not need detailed knowledge of REST for using the web service. The structure of a request and the replies delivered by the web service are explicated below.

Client and server communicate via HTTP, version 1.1.

4.1 System Requirements

The system on which you intent to install the NAST web service must satisfy the following requirements:

- You will need a Servlet container on which Servlet-API 2.4 has been implemented (e.g. Tomcat, version 5.5.x or higher),
- Java Runtime Environment, version 1.7 or higher,
- access to a default resolver,
- IPv4 or IPv6 access to the DNS.

4.2 Initial Installation

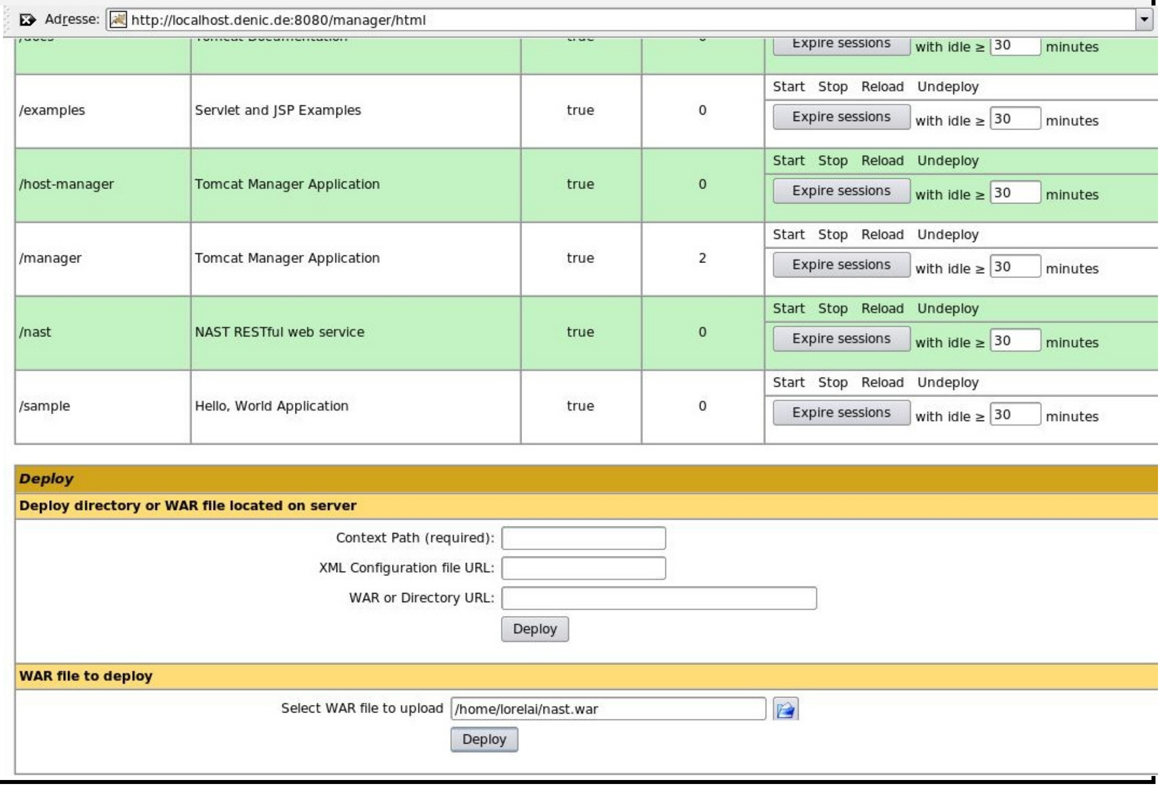
1. Depack the `nast.war` file provided in the archive.
2. Edit the `web.xml` file contained in `nast.war` (see chapter 4.6) and save the modified `web.xml` in the WAR file.
3. Then deploy the WAR file in the Servlet container or move the WAR file to the appropriate directory of the Servlet container and start the container again.

Example: Installation in case of Tomcat 6.0.20:

```
[lorelai@starshollow]$ cp nast.war ~/apache-tomcat-6.0.20/webapps/
[lorelai@starshollow]$ cd ~/apache-tomcat-6.0.20/bin/
[lorelai@starshollow]$ ./startup.sh
[lorelai@starshollow]$
```

[2] *http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm*

Example: Deployment in case of Tomcat 6.0.20 via the management console:



Adresse: http://localhost.denic.de:8080/manager/html

[illegible]	[illegible]	[illegible]	[illegible]	Expire sessions with idle ≥ 30 minutes
/examples	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	Tomcat Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	Tomcat Manager Application	true	2	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/nast	NAST RESTful web service	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/sample	Hello, World Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path (required):

XML Configuration file URL:

WAR or Directory URL:

Deploy

WAR file to deploy

Select WAR file to upload /home/lorelai/nast.war

Deploy

4.3 Start and Stop

The web service is started and stopped by means of the auxiliary means provided by the chosen Servlet container (such as start/stop scripts or management console).

4.4 Update of the Web Service

To update the web service, stop and undeploy the web application currently running in the Servlet container and install the service as described in chapter 4.2.

Please note that you must re-configure the parameters in the `web.xml`.

4.5 Deinstallation

To de-install the software, you may either stop the Servlet container and delete the application or you may undeploy the service.

4.6 Configuration

The `web.xml` file offers the following configuration parameters:

Parameter	**Meaning**	**Value Range**	**Mandatory / Optional**
default.resolver.interface	Host name of default resolver *(Default: localhost)*	Name of interface or IP address	optional
default.resolver.port	Port number of default resolver *(Default: 53)*	0 – 65535	optional
global.unicast.ipv6.url	URL for loading the "IPv6 Global Unicast Address Assignments" *(Default: http://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xml*) Please note that the URL can also be of type file://<file_name>	Valid URL in accordance with RFC3986	mandatory

If you do not configure a default resolver, NAST will try to automatically determine the default resolver of the platform. The application will follow the procedure documented at:

http://www.dnsjava.org/dnsjava-current/doc/org/xbill/DNS/ResolverConfig.html

4.7 Logging

The logging of the application can be configured via the `log4j.properties` file in the `WEB-INF/classes` directory.
According to the default setting, the web service logs in the `nast.log file`.
Each request is logged in the following format:

```
timestamp, second since epoch|policy token|quell-ip-addresse|
Laufzeit, Millisekunden, ganzzahlig|NAST-Status|
Liste der NAST-Warnungscodes|
```

This data can be analyzed, e.g. for statistical purposes.

4.8 HTTP Request

A HTTP request is composed of a Uniform Resource Identifier (URI) and a HTTP method. The URI identifies the resource. The HTTP method indicates what to do with the resource.

4.8.1 URI

In the present context, you must state a domain name and at least 2 name servers (and their glue addresses as optional data) in the URI. Additionally, the URI must contain information about the display format of the reply data.

The syntax of a valid URI reads as follows (variable parts of information in <pointed brackets>):

http://<host name>:<Port>/nast/<display format>/<Domain>?ns1=<name server-1>&ns2=<nameserver-2>&nsX=<name server-N>...&nsX=<name server-N>

Note: The third and all further name servers *always* have the URL parameter nsX. This means the line is not continued with ns3, ns4 and so forth.

The components of the URI:

Variable	**Meaning**	**Potential values**	**Mandatory / Optional**
Host name	Name of the host on which the web service is running.	Local host, name of interface or IP address	mandatory
Port	Port number of the host on which the web service is running.	0 – 65535	optional
Display format	Format in which the client requests the response to be displayed. Possible formats are XML or JSON.	XML[3] JSON[4]	mandatory
Domain	Name of the domain that is to be checked	Valid domain	mandatory
ns1-nsX	Names (and optionally glue addresses) of the name servers to be checked.	Format: Host name[,IPv4[,IPv6]] You may use IPv4 as well as IPv6 addresses as glue addresses.	Mandatory: at least two name servers

[3] http://de.wikipedia.org/wiki/XML

[4] http://de.wikipedia.org/wiki/JSON

Example: URI

The reply shall be displayed in XML notation (identifier "xml" in the URI) :
http://host.example:8080/nast/xml
/example-eins.de?ns1=ns1.example-
eins.de,81.91.170.1,2001:608:6:6:0:0:0:11&ns2=ns2.denic.de

The reply shall be displayed in JSON notation (identifier "json" in the URI) :
http://host.example:8080/nast/json
/example-eins.de?ns1=ns1.example-
eins.de,81.91.170.1,2001:608:6:6:0:0:0:11&ns2=ns2.denic.de

In the above example, an HTTP request is addressed to the host "host.example" on port 8080. By stating "nast" in the URI, the host realizes that the Documentation shall be called. It shall be checked whether the delegation of the "example-eins.de" domain to the name servers "ns1.example-eins.de" and "ns2.denic.de" satisfies the name server delegation requirements of DENIC.

4.8.2 HTTP Methods

The web service allows for the following HTTP methods: GET, HEAD and OPTIONS.

- GET is used to execute the Nameserver Predelegation Check.
- HEAD delivers the same HTTP header as GET, but without the reference data in the body.
- OPTIONS informs the client about the HTTP methods that are supported by the server.

4.8.3 HTTP Response

The web service answers with a HTTP response, which is composed of a message head and a message body, as it is standard for HTTP. In the message head, the HTTP status code and various HTTP headers are stored. The HTTP headers supply information such as meta information about the body (e.g. which type of encoding is used). The body contains the reference data, in the present case the result of the Nameserver Predelegation Check.

4.8.4 HTTP Status Codes

The table below gives the HTTP status codes that may be used:

Status Code	Status Message	Explanation
200	OK	The request was successful.
400	Bad request	The server was unable to read the request due to its faulty syntax.
405	Method not allowed	The HTTP method stated in the request is not permitted.

500	Internal server error	An unexpected server error has occurred. The request could not be processed.

The HTTP status codes are defined in the RFC2616[5].

[5] http://www.ietf.org/rfc/rfc2616.txt

4.8.5 HTTP Header

You may use the following values as content-type HTTP header:

- application/xml;charset=ISO-8859-1: The content of the body will be displayed in XML notation.
- application/json;charset=ISO-8859-1: The content of the body will be displayed in JSON notation.

4.8.6 HTTP Body

The body contains the technical reply of the server.

Example: http body in JSON notation

{"success":false,"issues":[{"severity":"warning","code":120,"message":"Recursive queries should not
be allowed (resolver)","arguments":["/81.81.170.12:53
(UDP, Timeout: 5s, Retry: 3 x 5s)"]},{"severity":"error","code":116,"message":"Authoritative answer regarding SOA record is required (resolver, answer)","
arguments":["/81.81.170.12:53 (UDP, Timeout: 5s, Retry: 3 x 5s)","/ 81.81.170.12:53 (UDP, Timeout: 5s, Retry: 3 x 5s): RA <No records>"]},{"severity":"er
ror","code":116,"message":"Authoritative answer regarding SOA record is required (resolver,
answer)","arguments":["/81.81.170.12:53 (UDP, Timeout: 5s, Retry
: 3 x 5s)","/ 81.81.170.12:53 (UDP, Timeout: 5s, Retry: 3 x 5s): RA <No
records>"]},{"severity":"warning","code":1020,"message":"Recursive queries should not
be allowed (resolver)","arguments":["/81.81.170.12:53 (UDP, Timeout: 5s, Retry: 3 x 5s)"]}]}

Example: http body in XML notation

```
<?xml-stylesheet type="text/xsl" href="../meta/nast_answer.xsl"?>
<predelegation-check success="false" xsi:schemaLocation="../meta/nast_answer.xsd"
xmlns="http://schema.denic.de/rng/nast" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
<issues>
<issue code="120" severity="warning">
<message>Recursive queries should not be allowed (resolver)</message>
<parameters>
<parameter>/81.81.170.12:53 (UDP, Timeout: 5s, Retry: 3 x 5s)</parameter>
</parameters>
</issue>
<issue code="116" severity="error">
<message>Authoritative answer regarding SOA record is required (resolver,
answer)</message>
<parameters>
<parameter>/81.81.170.12:53 (UDP, Timeout: 5s, Retry: 3 x 5s)</parameter>
<parameter>/81.81.170.12:53 (UDP, Timeout: 5s, Retry: 3 x 5s): RA &lt;No
records></parameter>
</parameters>
</issue>
<issue code="106" severity="error">
<message>Authoritative answer regarding SOA record is required (resolver,
answer)</message>
<parameters>
<parameter>/81.81.170.12:53 (UDP, Timeout: 5s, Retry: 3 x 5s)</parameter>
<parameter>/81.81.170.12:53 (UDP, Timeout: 5s, Retry: 3 x 5s): RA &lt;No
records></parameter>
</parameters>
</issue>
<issue code="120" severity="warning">
<message>Recursive queries should not be allowed (resolver)</message>
<parameters>
<parameter>/81.81.170.12:53 (UDP, Timeout: 5s, Retry: 3 x 5s)</parameter>
</parameters>
</issue>
</issues>
</predelegation-check>
```

The reply contains certain keywords to which specific values are assigned:

Keyword	Meaning	Possible values and their meaning
success	Indicates whether the Nameserver Predelegation Check was successful.	true (The check was successful.) false (The check was not successful, there is at least one issue marked with severity=error.)
issues[6] (list)	A list of error messages. It lists the recommendations or mandatory requirements described in the document DENIC-23 – Nameserver Predelegation Check that have not been met.	1 – n error messages. Every error message (issue) has the following attributes (maximum): severity (mandatory) code (mandatory) message (optional) parameters (optional).
severity	Weighting	Warning Error
code	Unique error code	Numeric code
message	Explanatory text	String
parameters	0 – n parameters. A parameter states which object is referenced by the error message.	String
arguments	Is used in case of a JSON notation instead of "parameters", remaining keywords are identical.	String

4.9 XML Schema Files

You can use the relative URIs listed below to download the schema files belonging to the XML notation:

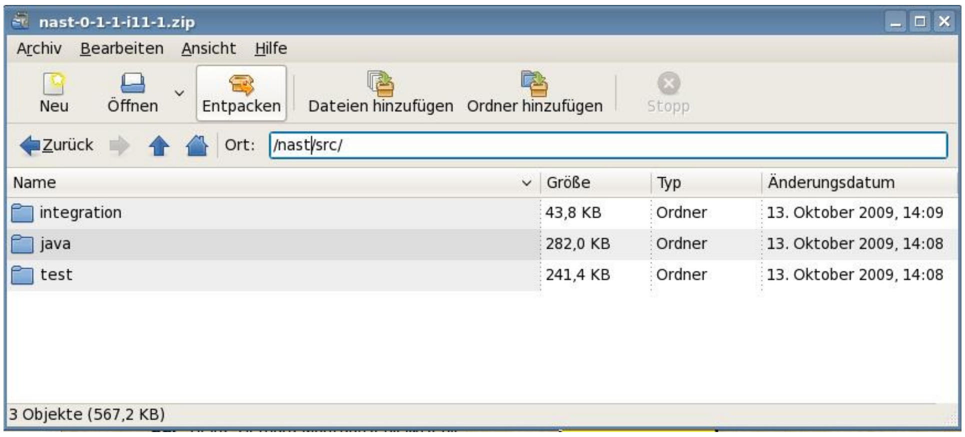
Schema	Relative URI
RELAX NG schema[7]	/nast/meta/nast_answer.rng
XML schema	/nast/meta/nast_answer.xsd

[6] see chapter 7. Issue Codes

[7] See http://relaxng.org/

5. Program Source Text

You will find the program source text of NAST in the *src* directory:


nast-0-1-1-i11-1.zip
Archiv Bearbeiten Ansicht Hilfe
Neu Öffnen Entpacken Dateien hinzufügen Ordner hinzufügen Stopp
Zurück
Ort: /nast/src/

Name	Größe	Typ	Änderungsdatum
integration	43,8 KB	Ordner	13. Oktober 2009, 14:09
java	282,0 KB	Ordner	13. Oktober 2009, 14:08
test	241,4 KB	Ordner	13. Oktober 2009, 14:08

3 Objekte (567,2 KB)

The source text gives you the possibility to learn how the predelegation checks are executed. You may integrate the source text into your own application. Please note that we will not distribute the source code of the libraries developed by DENIC. But we will make available to you the compiled libraries. This will enable you to compile the NAST source code. You will find the related JAR files in the WEB-INF/lib directory of the NAST web service implementation.

In the delivered Javadoc you will find the central functional entry point at the *de.denic.zone.nast.. business.PredelegationCheck* interface with its implementing classes, in particular PredelegationCheckImpl.

The first of the four constructors of this class is the easiest one to use. It depends on three object instances:

a) An ExecutorService from JDK

b) An implementation of the de.denic.techsupport.dns.DnsQuery interface, which encapsulates the processing of DNS queries. You will find the implementation *de.denic.techsupport. dns.impl.DnsQueryDnsJavaImpl* in the JAR file *techsupport-dns-<VERSION>.jar*. This implementation is in turn based on the dnsjava library. Besides a default constructor, it offers the possibility of configuring the default recursive resolver. The default constructor uses the OS/VM stub resolver for this purpose. If you want to build your own implementation of the *de.denic.techsupport.dns.Resolver interface* we recommend using the *de.denic.techsupport. dns.ResolverBuilder interface*.

c) An implementation of the de.*denic.techsupport.ipaddress.IPv6GlobalUnicastAddressCheck* interface. This interface has the task to verify if a given IPv6 address belongs to an address block allocated by IANA. You will find the implementation *de.denic.techsupport.ipaddress. impl.IPv6GlobalUnicastAddressCheckIanaPageParsingImpl* in the JAR file *techsupport-ipaddress-<VERSION>.jar*. To be able to use this implementation, you need a URL that points to the text or XML source where one can get the required information.

Additionally, you must provide a flag indicating the result that should be delivered in case that an IPv4 address (instead of an IPv6 address) is erroneously passed on for verification.

You may optionally pass on a ScheduledExecutorService as an argument to define the actualization frequency of the IANA address blocks from the configured URL.

If you want to make any suggestions or proposals for improvements, you are welcome to send them to *nast-feedback@denic.de*.

6. Issue Codes

At present, the following defined issue codes exist:

Issue-Code	**Message**	**Reference to chapter / paragraph in the documentation DENIC-23**
101	Missing glue record for the nameserver (NS)	Chapter 2.1.3, paragraph 1a
102	Provided glue records not applicable (NS)	Chapter 2.1.3, paragraph 1b
103	Nameservers having IPv6 glue records should have IPv4 glue records too (NS, # of IPv4 glues, # of IPv6 glues)" RECURSION_AVAILABLE => "Recursive queries should not be allowed (resolver)	Chapter 2.1.3, paragraph 2
104	Calculated referral response larger than allowed (length in octets)	Chapter 2.1.5, paragraph 3
105	All IPv6 Addresses must be Global Unicast dedicated, allocated and routable	Chapter 2.1.6, paragraph 1
106	Inconsistent set of name server IP addresses (NS, provided glues, determined glues)	Chapter 2.1.3, paragraph 3
106	Received response does not provide expected records directly (resolver, NS, RR)	Chapter 2.1.3, paragraph 3
106	Received response not authoritative (resolver, NS, RR)	Chapter 2.1.3, paragraph 3
107	Insufficient diversity of nameserver' s IP addresses (expected, found)	Chapter 2.1.2
107	Insufficient diversity of nameserver' s IPv4 addresses (expected, found, IPs found)	Chapter 2.1.2
107	Insufficient number of nameservers reachable via IPv4 (expected, found)	Chapter 2.1.2
107	Insufficient number of nameservers reachable (expected, found)	Chapter 2.1.2
108	Refresh value out of range	Chapter 2.1.4, paragraph 2a
109	Retry value out of range (expected, found)	Chapter 2.1.4, paragraph 2b
110	Retry value out of range (expected, found)	Chapter 2.1.4, paragraph 2c
111	Expire value out of range (expected, found)	Chapter 2.1.4, paragraph 2d

Issue-Code	**Message**	**Reference to chapter / paragraph in the documentation DENIC-23**
112	Minimum TTL out of range (expected, found)	Chapter 2.1.4, paragraph 2e
113	Primary Master (MNAME) inconsistent across SOA records	Chapter 2.1.5, paragraph 4
114	Inconsistent serial number across SOA records (serial number)	Chapter 2.1.4, paragraph 1
115	SOA record response must be direct (resolver, answer)	Chapter 2.1.5, paragraph 2
116	SOA record response must be authoritative (resolver, answer)	Chapter 2.1.1
118	Inconsistent set of NS RRs (IP, NS host names)	Chapter 2.1.5, paragraph 1
118	NS query response is empty	Chapter 2.1.5, paragraph 1
119	Some Nameservers not reachable via TCP	Chapter 2.1.6, paragraph 3
120	Recursive queries should not be allowed (resolver)	Chapter 2.1.6, paragraph 2
121	Received a truncated response (resolver, answer)	No Chapter
200	DNSKEY RR ZONE flag (bit 7) must be set	Chapter 3.6.1.1, paragraph 1
201	DNSKEY RR REVOKE flag (bit 8) must not be set	Chapter 3.6.1.1, paragraph 2
202	DNSKEY RR SEP flag (bit 15) should be set	Chapter 3.6.1.1, paragraph 3
203	DNSKEY RR RSA key modulus length in bits out of range	Chapter 3.6.1.4, paragraph 1
204	DNSKEY RR RSA public key exponent length in bits must not exceed 128 bits	Chapter 3.6.1.4, paragraph 2
205	DNSKEY RR DSA public key parameter T out of range	Chapter 3.6.1.4, paragraph 3
206	DNSKEY RR DSA public key has invalid size	Chapter 3.6.1.4, paragraph 4
207	DNSKEY RR public key must be BASE64 encoded	Chapter 3.6.1.4
208	Duplicate DNSKEY RR	Chapter 3.6.1
209	At least one DNSKEY RR must be specified in request	Chapter 3.6.1.4
210	At most 5 DNSKEY RR allowed	Chapter 3.6.1, 3.6.2
211	Inconsistent DNSKEY RR in nameserver response	Chapter 3.6.2

Issue-Code	**Message**	**Reference to chapter / paragraph in the documentation DENIC-23**
212	Did not find DNSKEY RR from request in nameserver response	Chapter 3.6.2, paragraph 2
213	No DNSKEY RR from request found in nameserver response	Chapter 3.6.2, paragraph 2
214	Some nameservers not reachable via EDNS0 with sufficient packet size	Chapter 3.6.5, paragraph 2
215	Timeout after switching from UDP to TCP - switch to TCP due to truncation	Chapter 3.6.5, paragraph 3
216	No visible DNSKEY found signing the DNSKEY RR obtained in response	Chapter 3.6.3
217	No visible DNSKEY found in signing directly or indirectly the SOA RR obtained in response	Chapter 3.6.4
218	Received invalid answer to a DO-Bit query	Chapter 3.6.5, paragraph 2
219	Unable to retrieve DNSKEY RR with TCP or EDNS0	Chapter 3.6.5, paragraph 3
220	DNSKEY RR has invalid algorithm	Chapter 3.6.1.4., paragraph 4
221	Unknown flags in DNSKEY RR are set	Chapter 3.6.1.1
222	Some nameservers not reachable via EDNS0 because of occurred timeout	Chapter 3.6.5, paragraph 2
223	Timeout after switching from UDB to TCP - switch to TCP due to timeout	Chapter 3.6.5, paragraph 3
224	Some nameservers not reachable via EDNS0	Chapter 3.6.5, paragraph 2
225	Timeout after switching from UDP to TCP	Chapter 3.6.5, paragraph 3
226	DNSKEY RR ECDSA public key has invalid size	Chapter 3.6.1.4, paragraph 5 Chapter 3.6.1.4, paragraph 6
227	DNSKEY RR GOST public key has invalid size	Chapter 3.6.1.4, paragraph 7
901	Unexpected RCODE (target, entity, RCODE)	No reference to any chapter. The RCODE of the reply is not NOERROR.

Issue-Code	**Message**	**Reference to chapter / paragraph in the documentation DENIC-23**
902	Timeout	No reference to any chapter. Timeout of communication with stated server.
903	Timeout with recursive resolver	No reference to any chapter. Internal Timeout – Please try again
904	Port unrechable	No reference to any chapter. Target nor listening at UDP Port 53
905	Invalid DNSKEY RR public key - conversion problem	Conversion of DNSKEY RR failed - Please contact Business Services
906	Invalid DNSKEY RR DSA public key - conversion problem	Conversion of DNSKEY RR failed - Please contact Business Services
907	DNSKEY RR from nameserver response cannot be compared with DNSKEY RR from request - conversion problem	Conversion of DNSKEY RR failed - Please contact Business Services
908	TCP connection refused	No reference to any chapter. TCP connection refused by server
909	Socket error	No reference to any chapter. Socket error, e.G. because the Network is unrechable
999	Unexpected exception	No reference to any chapter. This code means that the DNS query failed. Please contact Business Services.

7. Document History

Version	Date	Chapter	Amendment
0.1	17 Nov 2009	<all>	Document prepared
0.2	23.11.2009	3.3 & 4.6	Updated URL
0.3	24.11.2009	<all	Document updated
0.4	30.11.2009	3.4.2 & 4	Typos
0.5	12.01.2010	1.1, 6, 3.4.5, 3.4.6, 5	Update for NAST Version 1.0.0
0.6	22.01.2010	4.7	Added Chapter
0.7	29.01.2010	6	Added Issue-Code 120
0.8	10.02.2010	6	Typo: Issue-Code 107, removed Issue Code 117
0.9	23.02.2010	6	Added Issue-Codes 200 - 206
1.0	28.04.2010	1.2 4.8.1 3.4.6 3.4.7	Updated NAST-Version Added Policy and Dnskey Chapter updated Chapter added
1.1	14.06.2010	6	Added Issue-Codes 200-225, 905-907
1.2	06.07.2010	6	Description Issue-Code 103 and 105 Added Issue-Code 904
1.3	16.09.2010	1.1, 1.3. 4.6.1	Document updated for ENUM
1.4	23.5.2010	6	121: Received a truncated response (resolver, answer) 207: Kap. 3.6.1.4 Absatz 4 -> Kap. 3.6.1.4 208: Kap. 3.6.5 -> Kap 3.6.1 209: Kap. 3.6.1.4 Absatz 4 -> Kap. 3.6.1.4 210: Kap. 3.6.5 -> 3.6.1 und 3.6.2 212: Kap. 3.6.2 -> Kap 3.6.2 Absatz 2 213: Kap. 3.6.2 -> Kap 3.6.2 Absatz 2 215: -> Timeout after switching from UDP to TCP - switch to TCP due to truncation 218: Kap. 3.6.2 -> Kap. 3.6.5 Absatz 2 223: -> Timeout after switching from UDP to TCP - switch to TCP due to timeout
1.5	20.6.2011	3.4	Modifications in the CLI options:Since v1-3-2 (Refer to NAST-55) 1) '-ri' has become '-i' 2) '-p' has become '-D' 3) '-rp' has become '-p' 4) '-?' dropped ('-h' für Help remaining)
1.6	20.6.2011	3.4.3	Added: Parameter „`-p <Port>`“ is mandatory.
1.7	10.03.2014	6	Updated Issue-Code 107

1.8	01.12.2014	3.1, 4.1, 4.7	Updated system rquirements and loggig informations
1.9	20.08.2015	6	Added Issue-Codes 226 and 227