FELICE Netgate2 Protocol Specification

Release 0.4

Gwendolyn van der Linden
FOM Institute for Plasma Physics 'Rijnhuizen'
Edisonbaan 14
3439 MN Nieuwegein
The Netherlands

June 3, 2008

Contents

List of Figures

List of Tables

Chapter 1

Scope

1.1 Identification

This document relates to the second generation *netgate* protocol, netgate2, to be used for communication between and with the FELICE control and diagnostic subsystems. It is based on the netgate protocols used in FELIX.

1.2 System overview

FELICE is the planned extension to the FELIX setup at FOM Rijnhuizen [1, 2].

1.3 Document overview

This document is aimed at designers and implementors of FELICE subsystem software and designers and implementors of netgate2 client libraries. It provides the background, specification, design and high level implementation of the netgate2 protocol, and the network layout.

1.4 Change history

1.4.1 Releases

The document versions and their release dates are:

Version	Date
0.1	April 15, 2004
0.2	August 6, 2004
0.3	April 8, 2005
0.4	May 12, 2005

Note: attributes, decisions, *etc.* are numbered in order of appearance, and can be renumbered due to addition of removal of requirements. The numbers used in the following sections always refer to the current numbering, as used in this document.

1.4.2 From release 0.1 to 0.2

The following changes were made to the specification:

1. added attribute ATTR-2;
2. added decision DEC-2.

The following additions were made to this document:

1. a new chapter on transition planning;
2. a new chapter on implementation.

1.4.3 From release 0.2 to 0.3

The following changes were made to the specification:

1. strings are always prefixed by a length parameter (not quoted, as previously);
2. the message format is extended with a protocol version, data format indicator, and return error code;
3. binary data is now allowed in the message body, when setting the data format indicator to a non-ASCII type;
4. added decision DEC-10;
5. extended decision DEC-8 with supervisor concept;
6. added item 4 to decision DEC-4;
7. error responses may be ASCII, even if the command is not;
8. added BNF specification of message format to appendix.

1.4.4 From release 0.3 to 0.4

The following clarifications were made to the specification:

1. the strings length parameter is always followed by a space;
2. non-exponential decimal point representation of floating point number is allowed, so an example has been added to stress that;
3. moved FEL-number requirement to FELICE chapter;
4. fixed small textual errors.

Chapter 2

Current situation

2.1 FELIX system overview

The current network topology for FELIX is depicted in Figure 2.1.

2.2 Existing netgate protocols

It is desirable for the netgate2 protocol to be similar to the existing netgate protocols. That way it is possible to build on existing experience and minimize the effort needed for implementing the new protocol. The existing FELIX netgate protocols have the following properties:

- short-lived TCP/IP connections;
- ASCII strings to encode the messages;
- command-response message pairs.

The command string does not have a length field, as it is assumed to be sent in a single TCP data packet. The response string from the diagnostic subsystem has a fixed length header containing the status code field, and length field, indicating the length of the response payload. The response string from the other FELIX subsystems has a fixed length header with the status code field, and a fixed length payload with the response data. The length of this response string can be different for different types of commands.

For all subsystems the status code field has a width of 2 bytes, where `00` is used to indicate success, and `-1` to indicate failure. The spectrum analyzer commands use round brackets to indicate the command parameter, while the commands for the other subsystems use space delimited command arguments. All subsystems send responses with space delimited data fields.

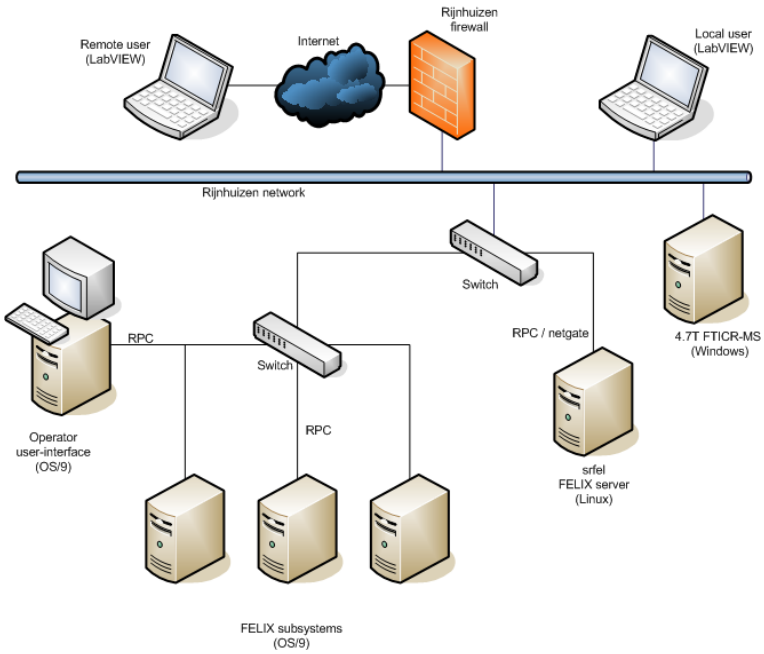

Figure 2.1: FELIX network topology

Chapter 3

FELICE communication requirements

The planned network topology for FELICE is depicted in Figure 3.1.

Required attribute 1 *The netgate2 protocol shall be sufficiently fast to sustain the worst-case data-flow between all the FELICE and FELIX subsystems, using a* 100 *Mbit UTP network.*

The worst-case situation is estimated to be:

- 20 Hz update frequency;
- 4 subsystems, 1 netgate server, 1 operator user-interface, 5 user clients;
- 10 messages per subsystem;
- worst-case average of 20 scalar (floating-point) values per message.

This amounts to roughly 1000 messages per second with 200 bytes per message, or approximately 2 Mbit/s. Given the 100 Mbit/s network, the data throughput is not expected to be a problem, but the latency and overhead of individual messages may need special attention.

Required attribute 2 *The netgate2 protocol shall allow for sufficiently fast detection and handling of error situations.*

FELIX will operate at 20 Hz pulse trains. It is considered to be sufficient to react on an error situation within one or two pulses (hence 50 to 100 ms).

Constraint 1 *The FELICE subsystems shall be implemented in LabVIEW Real-Time.*

This decision has already been made in an earlier stage of the project.

Constraint 2 *The netgate2 server shall be running on a Linux PC, and be implemented in Java.*

The current netgate server is implemented in C, which is still considered to be a good choice. However, an important part of the netgate server functionality deals with socket communication, multi-threading, error handling, and message logging, which can be handled more easily in Java using standard libraries and components.

Constraint 3 *The operator user-interface shall be implemented in LabVIEW.*

This operator user-interface will, in time, be extended to replace the operator user-interface for FELIX. The current operator user-interface for FELIX is implemented in GWindows running on OS/9. LabVIEW is considered to be a better choice, providing an improved user-interface with less effort.

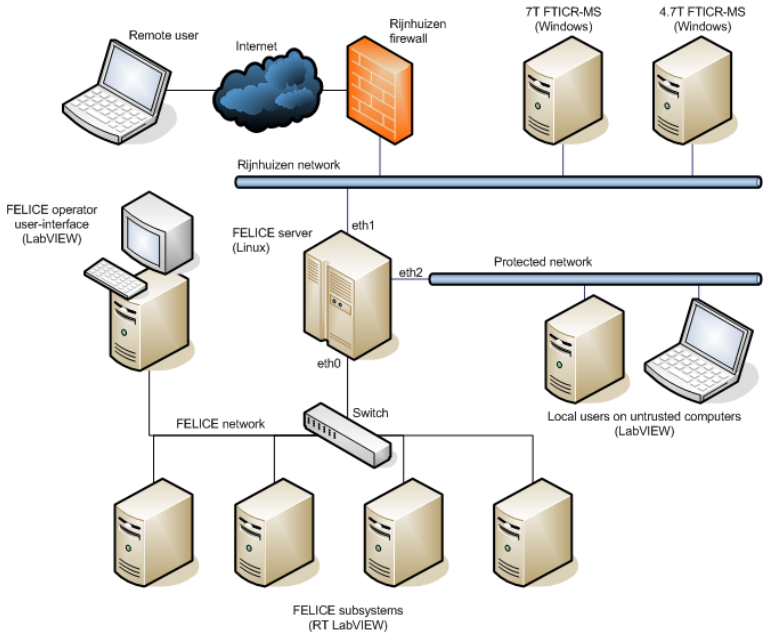

Figure 3.1: FELICE network topology

Required attribute 3 *The user clients shall not be able to block the operator from accessing FELICE through the operator user-interface.*

The operators must be able to access FELICE at all times. The user may, willingly or unwillingly, attempt to monopolize the FELICE system, for example by failing to release network connections.

Constraint 4 *The netgate2 client interface shall be implemented in C, Java and in LabVIEW.*

Most FELICE users will be using LabVIEW to interface their system, and will want to integrate interfacing FELICE into their application. Some FELICE users, such as the FELICE/FTICR-MS users, will use a Microsoft Windows application that must interface FELICE using C/C++.

Functional requirement 1 *Access to FELICE shall be controlled, allowing different settings for:*

- *read access,*
- *operator write access, and*
- *user write access.*

The access control functionality must allow:

- *limiting access to specific users, based on IP-address and username/password;*
- *revoking granted access permissions.*

Typically, everybody may gain read access to FELICE, while operators may gain write access to all subsystems. Designated users may gain write access to a subset of the FELICE functions for a specific period of time.

Chapter 4

Netgate2 design

In this chapter the design of the overall netgate2 system, and the netgate2 protocol format is described.

Decision 1 *All external client access to FELICE, and corresponding access control shall be handled by the netgate2 server. The FELICE subsystems are placed on a subnet (see Figure 3.1), and are not directly accessible from the Rijnhuizen intranet, extranet, or the Internet.*

Consequently, the subsystems do not need to implement access control, and can use a single service that handles all read and write commands by both operators and users. For debugging purposes, the subsystems can be accessed directly from a computer on the subnet.

Decision 2 *The netgate2 server is configured with netfilter (`http://www.netfilter.org`) to provide access control based on adapter (`eth0`,`eth1`), ip-address and port number.*

This provides a firewall.

Decision 3 *Read commands, operator read/write commands, and user read/write commands are each handled on a different port on the netgate2 server. The netgate2 server will direct the valid requests to the corresponding subsystem.*

This simplifies access control, as most of it can be covered by configuring the firewall on the netgate2 server. For example, the operator read/write port can be restricted to be accessible only from the local FELICE network.

Decision 4 *The netgate2 protocol follows these rules:*

1. *all read and write functions are implemented as command-response message pairs;*
2. *each subsystem regularly broadcasts status information (formatted as a response message) on the subsystem subnet using UDP multicast;*
3. *detected error conditions are broadcasted immediately, and are included in the regular status message for as long as the condition exists;*
4. *subsystems can send one-way messages to the server using UDP or TCP;*

The command-response messaging provides a simple and reliable communication channel. The broadcast allows for the timely distribution of information across subsystems, and avoids the need for polling. The one-way messaging allows for the timely distribution of information to the server, without the need for polling. One-way UDP messages can be used when resending a lost message is not desired (*e.g.* it may be better to send a new up to date message). One-way TCP can be used for reliable delivery of messages.

Type	Len	Field
int	6	payload length
token	?	command
int	?	protocol version
char	1	data format
?	?	data

Figure 4.1: Command message format

Type	Len	Field
int	6	body length
token	?	command
int	?	protocol version
char	1	error group
int	?	error number
int	1	error level
int	?	error text length
text	?	error text
char	1	data format
?	?	data

Figure 4.2: Response / one-way message format

Decision 5 *The netgate2 server checks if each subsystem sends broadcast messages. When a time-out threshold is exceeded, the netgate2 server takes action, depending on the criticality of the subsystem. If a critical subsystem stops broadcasting, FELIX/FELICE is shutdown. Failure of a non-critical subsystem just generates a warning that is displayed on the operator console.*

When defining the message format, the following nomenclature is used:

length field is the fixed size part of the message indicating the payload length;

payload is the variable size part following the length field, and can be split into two parts: the header and the body;

header is the first part of the payload, having a fixed format, but variable size;

body is the second part of the payload, containing the message arguments.

int	integer value, left aligned where appropriate, not zero prefixed
token	ASCII string taking characters from `A-Za-z0-9_`
char	single 7-bit ASCII character, excluding space (code 32) and lower
text	ASCII string with 7-bit characters, excluding NULL.
data format	`A`: ASCII, `F`: LabVIEW flattened
error group	`L`: LabVIEW error code, `F`: FELICE application error code
error level	`0`: no error, `1`: warning, and `2`: error

Table 4.1: Message format definition vocabulary

command	19	`sa_spectrum_get 1 A`
response	112	`sa_spectrum_get 1 F 0 0 A 0.0 0.1 0.5 ...`
command	23	`oc_vas_status_get 1 A 4`
response	29	`oc_vas_status_get 1 F 142 2 A`
command	17	`uc_scan_start 1 A`
response	25	`uc_scan_start 1 F 0 0 A`

Table 4.2: Examples of commands and responses

Decision 6 *The netgate2 messages must be formatted according to Appendix A (see also Figures 4.1 and 4.2). Additionally:*

1. *all message headers are string based, limited to the 7-bit ASCII character set*[1]*, and are **not** zero-terminated;*
2. *the body of an ASCII message is limited to the 7-bit ASCII character set, and is **not** zero-terminated;*
3. *both command and response start with a 6-byte left-aligned decimal length field, indicating the payload length in bytes, followed by a space, making it a 7 byte field;*
4. *all fields are space delimited, including the length field;*
5. *the space after the length field is considered to be part of the payload, and must NOT be counted when determining the payload length;*
6. *both the command name field in a command, and the data label field in a response shall start with a 2-character prefix identifying the subsystem, and shall use underscore delimited words;*
7. *all command names end in a verb stating the action, such as* `_set` *and* `_start`*;*
8. *to simplify access control (see FR-1), all read command names end in* `_get`*;*
9. *floating point numbers must be expressed using scientific decimal point notation (e.g.* `3.14e+5` *and* `1234567.89`*), where the required resolution is specified for each command separately;*
10. *strings are prefixed by a string length field (e.g. the error text in the response message); the space after the length field is always present, also when the string is empty and consequently has a zero length;*
11. *line breaks are expressed using a single* `\n` *character (UNIX convention);*
12. *a normal response must be in the same data format as the command, i.e. an ASCII command must be followed by an ASCII response;*
13. *an error response may be in ASCII, irrespective of the command type;*
14. *errors indicated in the response message must refer to the context of the command, and must not be used to return general error information of the subsystem (use status broadcasts for the latter).*

Some example messages are listed in Table 4.2.

Decision 7 *The netgate2 server must support simultaneous client connections on the same port.*

[1]This allows a future extension to the protocol: *e.g.* setting the highest bit of the first byte can be used to indicate a binary message.

This allows for long-lived connections, and prevents clients from monopolizing the subsystems.

Decision 8 *Subsystems shall not communicate with each other directly. Any information needed from other subsystems must be obtained by listening to status broadcast messages. Functionality that exceeds the scope of a single subsystem shall be implemented in a central supervisor module.*

This minimizes the interdependency of the subsystems, and concentrates the required interdependency in a single module.

Decision 9 *The netgate2 server buffers the broadcasted subsystem status messages. Any client requesting the status information from the netgate2 server will be sent a copy of the most recent status message.*

This protects the subsystems from a high client poll frequency.

Chapter 5

Transition planning

FELICE is an extension to FELIX, but will also include an upgrade of FELIX hardware, and close integration of FELIX and FELICE. In this chapter the transition to the final FELIX/FELICE system is described. The current FELIX system is depicted in Figure 2.1, and the planned FELICE system is depicted in Figure 3.1. FELIX must be operational as much as possible, also during the transition to the final FELIX/FELICE system.

Decision 10

The current `sa` *(spectrum analyzer) subsystem will not be upgraded to netgate2. The* `sa` *replacement, a new VME system, will not be connected to any of the subsystems, netgate server, or operator console in any way.*

The following approach will be used during the transition:

- Keep FELIX operational with the current hardware and software, for as long as needed.
- Keep FELICE subsystems, server, and user-interface separate from the FELIX system, as much as possible.
- Patch the old FELIX system where necessary to keep a working system, and keep the new system as clean as possible.

This results in the following plan:

1. Add subsystems for FELICE in parallel to the existing subsystems. This is not possible for the new beam optics subsystem, which must replace the old system, and support both FELIX and FELICE. Hence, the new beam optics subsystems will support FELIX/FELICE from the start, and completely replace the old subsystem.
2. Update `srfel` whenever a subsystem is changed or replaced, to keep a working system.
3. Install the new FELICE server and new operator user-interface, and extend them each time a new FELICE subsystem is added. The existing operator user-interface remains to be used for controlling FELIX. The new operator user-interface will be used for the beam optics of FELIX/FELICE as soon as the new beam optics subsystem is installed; the old beam optics user-interface will then no longer be usable.
4. Prepare the subsystems, FELICE server application and new operator user-interface to fully replace `srfel`.
5. Then, move all functionality to the new FELIX/FELICE system, and remove `srfel` and the obsolete subsystems.

Chapter 6

Netgate2 FELICE subsystem protocols

In this chapter the complete list is given of the netgate2 message formats for all FELICE subsystems. This chapter serves as the specification for the FELICE netgate2 implementations.

6.1 Common features

A number of subsystems differentiate between the beam line they operate on. The beam line is indicated by the FEL number.

Decision 11 *The FEL number, whenever present, shall be the first parameter field in the data block;*

The prefixes for the various functional groups are listed in Table 6.1. The division into functional groups largely correspond to the division into subsystems.

6.1.1 Status message

All subsystems and the server handle status message requests:

command	17	`<prefix>_status_get 1 A` , *or*
response	29	`<prefix>_status_get 1 F 0 0 A 7 offline`

The format of the status response message body is defined separately for each subsystem. The status should include all information that a client needs in normal operation to use the subsystem,

Prefix	Function group
`ac`	arc correction
`bo`	beam optics
`ds`	diagnostic system / software interlock
`lg`	message logger
`oc`	optical cavity
`sa`	spectrum analyzer
`su`	supervisor
`sv`	server functions
`tm`	timer
`uc`	undulator control

Table 6.1: Function group prefixes

0	Debugging
1	Informational
2	Warning
3	Error
4	Critical error

Table 6.2: Log levels

e.g. if the subsystem is operational, if it's busy executing a command, and include a list of often used variables.

6.1.2 Info message

All subsystems and the server handle an info message request, helping in identifying the subsystem.

command	15	`<prefix>_info_get 1 A` , *or*
response	47	`<prefix>_info_get 1 F 0 0 A 26 beam optics subsystem v1.1`

Note that both prefixed and non-prefixed commands are defined, as with the status message.

6.1.3 Logging

Subsystems may generate log messages for debugging or system monitoring purposes. Storing and accessing logs is best implemented on the server using readily available logging tools.

Required attribute 4 *Logging messages shall not be critical to the operation of a subsystem.*

Critical subsystem functionality must be provided as separate commands and important information must be included in status data. If a subsystem is unable to contact the log server, it should ignore the error and proceed with normal operation.

Decision 12 *Log messages sent by the subsystems include subsystem prefix, log level, and log message. They do not include a time stamp; a UTC time stamp is added by the logger.*

The available log levels are listed in Table 6.2. Note that only the logger clock must be kept synchronized.

lg_log_write

Command or one-way message to send a log message. The message arguments must be in ASCII, and are:

1. originating subsystem prefix,
2. log level (see Table 6.2), and
3. log message string (length plus message).

The log message string may contain newline characters. Returns (when used with command/response protocol): response with no arguments.

xx_log_level_set

Command that sets the minimum level log messages must have to be sent to the logger. The message argument must be in ASCII, and is:

1. log level (see Table 6.2)

Returns: response with no arguments.

Code	Description	Explanation
0	No error	Command succeeded.
1	Internal error	Unexpected error, such as a software error.
2	General error	Uncategorized error, see error text for more details about what is wrong.
3	Network error	Could not send command or receive response.
4	Illegal header	Message header not formatted correctly.
5	Illegal argument	Message body not formatted correctly, for example wrong data type.
6	Out of range	Argument out of range.
7	Subsystem unavailable	Subsystem not alive.
8	Command unknown	Given command is not known.
9	Permission denied	Client has insufficient privileges.
10	Illegal state	Command cannot be executed in current state.

Table 6.3: Error codes

6.2 IR beam control

6.2.1 Attenuator control

6.2.2 Cavity length control

6.2.3 Undulator control

6.2.4 Arc correction

6.3 IR beam diagnostics

6.3.1 Alignment beam splitter

6.3.2 Spectrum analyzer

6.3.3 Interlock status

6.4 Netgate2 server

6.4.1 Access control

6.4.2 System log

6.5 Error codes

A single list of error codes is maintained, covering common errors and subsystem-specific errors. The error codes are listed in Table 6.3. To simplify client programming, the netgate2 server can provide the clients with a textual description of the error messages. The default language is English (`en_GB`).

```
command   22    sv_error_msg_get 1 A 5
response  34    sv_error_msg_get 1 F 0 0 A 16 Illegal Argument
```

Chapter 7

Implementation

7.1 Server

The Netgate2 server is implemented in Java. The details of the Java-code implementation of the protocol are provided as a separate documentation set, generated from the sources using `javadoc`. This section covers the high-level implementation of the server.

7.1.1 Implementation principles

- Each client connection is handled in a separate thread: multiple clients can communicate with the server at the same time.
- Each subsystem connection is locked for the duration of a command-response pair: only one client can communicate with a particular subsystem at a time.
- One subsystem may provide one or more functional units. The current implementation is limited to a one-to-one mapping between subsystem hardware (a PXI crate) and a functional unit (*e.g.* beam optics). This may change at a later time.

7.1.2 Server configuration

The main server properties are read from `server.properties`, a Java resource file. Logging properties are currently also read from `server.properties`, but may be placed in a separate file. See http://logging.apache.org/log4j/.

7.1.3 Access control configuration

The server checks incoming commands against a set of access rules. Each user group has an associated port, so commands can be checked based on the port they were received from. Access control is configured in a separate file for each group: in *group*`-access.properties`. Each line in the file contains an `ACCEPT:` or `REJECT:` keyword, followed by a regular expression. The first rule that matches the command name with the regular expression is executed. As an example, take:

```
ACCEPT: \w+*_get
REJECT: oc_\w+
ACCEPT: \w+*_set
```

Here any command that ends in `_get` is accepted, all commands that do not end in `_get` but start with `oc_` are rejected, and all commands that do not start with `oc_` but end with `_set` are accepted. All other commands are rejected (the default rule).

7.2 Subsystems

To be determined.

Chapter 8

User guide

8.1 Server startup

To start the server, take the following steps:

1. Make sure the `CLASSPATH` environment variable is set, and includes the Log4j far file, *e.g.* `export CLASSPATH=/usr/share/java/log4j.jar`.
2. Check the configuration files `server.properties` and *`group`*`-access.properties`, and make sure it matches the server, client access, and subsystem configuration.
3. Check the firewall settings `iptables -L` and make sure it matches with the intended use of the server ports.
4. Run the file `server.sh`

8.2 Test subsystem startup

A dummy subsystem implementation is provided with the server. To start it, make sure the `CLASSPATH` is set (see above), and start the subsystem as `./subsys.sh` *`prefix`*, *e.g.* `./subsys.sh oc`. The test subsystem reads its configuration from the `server.properties` file, based on its prefix. Logging properties are defined in `subsys-`*`prefix`*`.properties`, *e.g.* `subsys-oc.properties`.

8.3 Test client

A simple test client implementation is provided with the server. To run it, make sure the `CLASSPATH` is set (see above), and run `./client.sh` *`host port command`*, *e.g.*

```
./client.sh localhost 4301 'oc_status_get'
```

Note that different ports are configured on the server, each for a different group of users, as defined in `server.properties`. Commands can be sent through different ports, to test the user group access policies.

Appendix A

Message format specification

```
; NetGate2 Augmented BNF specification - follows RFC #2234.
;
; Differences from RFC #2234:
;  - unless otherwise specified, quoted strings are matched case sensitively.

command = <body-length> <SP> <command-name> <SP> <proto-version>
<SP> <data-type> <SP> [ <data> ]

response = <body-length> <SP> <command-name> <SP> <proto-version>
<SP> <error-group> <SP> <error-code> <SP> <error-level>
<SP> <error-source> <SP> <data-type> <SP> [ <data> ]

body-length = 6<DIGIT>
command-name = <TOKEN>
proto-version = <DIGIT>; NetGate2 protocol version
data-type = "A" / "F" ; A=ASCII, F=LabVIEW Flattened
data = *<ANY>

error-group = "L" / "F" ; L=LabVIEW, F=FELIX/FELICE
error-code = <INT>
error-level = "0" / "1" / "2" ; 0=no error, 1=warning, 2=error
error-source = <INT> <SP> <STRING>

; basic data types
INT = *<DIGIT>
STRING = *( <VCHAR> / <SP> / <LF> )
TOKEN = *( <ALPHA> / <DIGIT> / "_" )

; used ABNF core rules (for reference only)
ANY = %x00-FF ; any 8-bit character
ALPHA = %x41-5A / %x61-7A ; A-Z / a-z
DIGIT = %x30-39 ; 0-9
LF = %x0A ; line feed
SP = %x20 ; space
VCHAR = %x21-7E ; visible (printing) characters
```

Bibliography

[1] G. Meijer, A.F.G. van der Meer, and G. von Helden. FELICE; a free electron laser for intra-cavity experiments. Technical report, FOM Institute for Plasma Physics 'Rijnhuizen', 2001.

[2] A.F.G. van der Meer, J. Oomens, and B. Redlich. FOM-programme in development of virtual laboratory netherlands; development of remote control for the FTICR-mass spectrometer at FELIX/FELICE. Technical report, FOM Institute for Plasma Physics 'Rijnhuizen', 2003.