YAMI4

Requirements

For YAMI4Industry, v.1.3.1

Table of Contents

Document scope

This document describes the architectural elements and a set of requirements related to the YAMI4Industry package. Considering that the YAMI4 library is intended for use as a reusable component and deployed as part of a separate and complete project, it makes sense to treat these requirements as low-level requirements, where high-level ones are to be defined respectively for the final system.

The document applies to the 1.3.1 version of the YAMI4Industry package.
See the following web site for general information on the project:

http://www.inspirel.com/yami4/industry.html

Architectural elements

All architectural elements of the YAMI4 library are software components that provide their API in terms of types and functions. There are no internal threads of execution and as such all those components are passive and exhibit no behaviour other than transitions triggered by function calls. The only exception to this are activities of the underlying network layer, but these are out of scope of this document.

Serializer

Serializer is a stateless set of services that is used by other parts of the system in order to pack data in byte buffers or unpack data from byte buffers according to the common YAMI4 protocol. Serializer can be used directly by the YAMI4 user for formatting and interpreting message payloads and by the agent for formatting and interpreting message headers.

Requirements for the serializer module are numbered as Y4_SR_*n*.

Socket

Socket is a file descriptor of a system-level resource used for UDP and TCP communication. Client-side sockets are contained and managed by channels (a single channel manages one client-side socket), whereas server-side (listening) sockets are managed by agents (a single agent manages one listening socket, although the listening socket need not be actively used if the given agent is not configured for receiving incoming connections).

Sockets are expected to support operations as defined by the POSIX standard, there are no YAMI4-specific requirements for them.

Input buffer

Input buffer is a byte array, managed by a single channel, where data received by the channel's socket is stored, perhaps by accumulating it piece-by-piece as it is received, until it is recognized as a complete message. Input buffer is a passive entity without any specific operations and has no YAMI4-specific requirements.

Output buffer

Output buffer is a byte array, managed by a single channel, where outgoing messages are stored in the serialized form and from where data is consumed, perhaps in a piece-by-piece manner according to the rate accepted by the underlying network, by sending it to the channel's socket. Output buffer is a passive entity without any specific operations and has no YAMI4-specific requirements.

Options

Options is an aggregate of fields that are used to configure various runtime parameters of the components managed by a single agent. Options can be initialized and modified by the user and provided when the agent is initialized.

Requirements for the options module are numbered as Y4_OPT_*n*.

Utility layer

Utility layer is a set of services that act as replacements for functions from the standard C library (which itself is not referred from the YAMI4 code) or that provide commonly needed operations on sockets or byte buffers.

Requirements for the utility layer are numbered as Y4_U_*n* for general-purpose utilities and Y4_NU_*n* for network-related utilities, but since utilities are not supposed to be directly accessed by the user, these requirements can be considered as derived requirements.

Channel

Channel encapsulates a single socket with its associated input and output buffers and manages all state transitions that are necessary to receive and send messages to and from any given remote endpoint. Multiple channels (up to some static limit) can be managed by a single agent and since the input and output buffers are distinct, data transmission can be managed in duplex-mode. Channels are created when the outgoing connection is established with remote endpoint or when a new connection is accepted by the listening socket.

Requirements for the channel are numbered as Y4_CHN_*n*, but since channels are not supposed to be directly accessed by the user, these requirements can be considered as

derived requirements.

Agent

Agent encapsulates all resources used for communication with remote endpoints - in particular, a single agent manages the listening socket and a set of channels. There can be multiple independent agents used in the running program, as long as the system-level resources (like port numbers used by listening sockets) are not conflicting with each other.

Requirements for the agent are numbered as Y4_AGN_*n*.

User

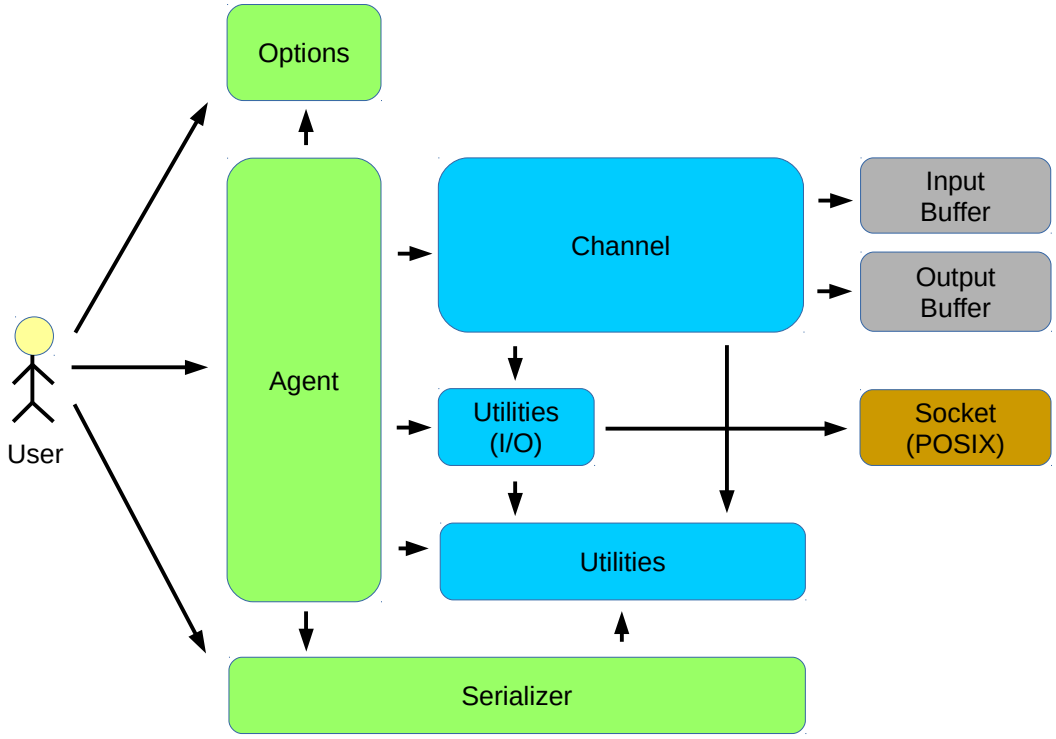
The user is a component (a running program) that instantiates one or more agents and invokes agent operations in order to manage external communication. The user can also directly refer to the serializer services and create and modify options.

General requirements

Apart from requirements that clearly belong to some architectural element, there are also requirements that specify concepts that are shared across the YAMI4 system. Such requirements are numbered as Y4_*n*.

Requirements relationships

The user, in general, interacts with one or many agents and uses the serializer facilities for preparing and interpreting messages – other functionalities are intended to be hidden from the user, as they are automatically invoked when needed. The following diagram presents the schematic relationships between architectural elements listed above and their respective requirements:



In this diagram:

- **GREEN** requirements represent public API of the YAMI4 library,
- **BLUE** requirements are derived and are intended for internal use, and
- **BROWN** requirements are assumed by reference to the POSIX standard.

Requirements

Y4_1

The TCP endpoint name has the following format:

```
tcp://a.b.c.d:p
```

where "a.b.c.d" is an IP address of the endpoint written in the numeric format and "p" is a TCP port number.

An example endpoint name, associated with the port 12345 on the local host, is "`tcp://127.0.0.1:12345`".

Y4_2

The UDP endpoint name has the following format:

```
udp://a.b.c.d:p
```

where "a.b.c.d" is an IP address of the endpoint written in the numeric format and "p" is a UDP port number.

An example endpoint name, associated with the port 12345 on the local host, is "`udp://127.0.0.1:12345`".

Y4_OPT_1

The yami_options_init function shall set all option fields to the values as described in Table 1.

Table 1

Option name	Initial value
`tcp_listen_backlog`	`10`
`tcp_reuseaddr`	`true`
`tcp_nonblocking`	`true`
`tcp_connect_timeout`	`0`
`tcp_nodelay`	`true`
`tcp_keepalive`	`false`

Note: *true and false are represented as 1 and 0 integer values, respectively.*

Y4_SR_1

The serializer shall return `yami_ok` when the requested operation was performed successfully.

Y4_SR_2

The serializer shall return `yami_not_enough_space` when there is not enough free space in the byte buffer or when there is not enough data remaining in the byte buffer to complete the requested operation.

Y4_SR_3

The serializer shall return `yami_unexpected_value` when the request to parse a given value was not successful due to invalid sequence encountered in the given buffer or when it was not possible to convert the encountered value to the given target type.

Y4_SR_4

The serializer shall pad the written value with trailing zeros up to the 4-byte alignment boundary when requested to write a value that has not aligned length.

Note: *This is particularly relevant with booleans, strings, binary sequences and their arrays, other types are naturally aligned. Note that the logical length of sequences is not affected by such padding.*

Y4_SR_5

The `yami_put_type` function shall convert type enumeration value to integer value as described in Table 2 and then serialize the integer value accordingly.

Table 2

Type name enumeration	Type name integer value
`yami_boolean`	1
`yami_integer`	2
`yami_long_long`	3
`yami_double_float`	4
`yami_string`	5
`yami_binary`	6
`yami_boolean_array`	7
`yami_integer_array`	8
`yami_long_long_array`	9
`yami_double_float_array`	10
`yami_string_array`	11
`yami_binary_array`	12
`yami_nested_parameters`	13
`yami_nested_parameters_array`	14

Note: *Y4_SR_6 covers serialization of integer values.*

Y4_SR_6

The `yami_get_type` function shall read the integer value and then convert it to type enumeration value as described in Table 2.

Y4_SR_7

The serializer shall serialize and deserialize 32-bit integer values in the given byte

buffer in the little-endian order.

Y4_SR_8

The serializer shall serialize and deserialize 64-bit long long values in the given byte buffer in the little-endian order.

Y4_SR_9

The `yami_put_integer` function shall place the 32-bit integer in the given byte buffer.

Note: *Y4_SR_7 specifies the requirement for serializing 32-bit integer values.*

Y4_SR_10

The `yami_get_integer` function shall read the 32-bit integer from the given byte buffer.

Note: *Y4_SR_7 specifies the requirement for deserializing 32-bit integer values.*

Y4_SR_11

The `yami_put_long_long` function shall place the 64-bit integer in the given byte buffer.

Note: *Y4_SR_8 specifies the requirement for serializing 64-bit long long values.*

Y4_SR_12

The `yami_get_long_long` function shall read the 64-bit integer from the given byte buffer.

Note: *Y4_SR_8 specifies the requirement for deserializing 64-bit long long values.*

Y4_SR_13

The `yami_put_double_float` function shall place the 64-bit float value in the given byte buffer, preserving the ordering of bytes.

Y4_SR_14

The `yami_get_double_float` function shall read the 64-bit float value from the given byte buffer, preserving the ordering of bytes.

Y4_SR_15

The `yami_put_cstring` function shall place the length of string (number of characters without any terminating or trailing values) as an integer value in the given byte buffer, followed by the sequence of characters, followed by any trailing zeros if needed for padding.

Note: *Y4_SR_4 specifies the requirement for padding for variable-length sequences.*

Y4_SR_16

The `yami_get_raw_cstring` function shall read the specified number of characters from the given byte buffer and terminate the sequence with the trailing zero.

Note: *the length prefix is not read, it is assumed that the user has already read the length prefix (as an integer value) before calling this function.*

Y4_SR_17

The `yami_put_binary` function shall place the length of byte sequence (number of bytes without any trailing values) as an integer value in the given byte buffer, followed by the sequence of bytes, followed by any trailing zeros if needed for padding.

Note: *Y4_SR_4 specifies the requirement for padding for variable-length sequences.*

Y4_SR_18

The `yami_get_raw_binary` function shall read the specified number of bytes from the given byte buffer.

Note: *the length prefix is not read, it is assumed that the user has already read the length prefix as an integer value before calling this function.*

Y4_SR_19

The serializer shall compress boolean arrays by storing or reading them in a packed form, with individual boolean values being assigned to consecutive bits of the bytes in the given byte buffer, where all bits are used with the possible exception of the last byte in the buffer, if the number of boolean values to be stored or read is not divisible by 8.

Note: *Y4_SR_20 specifies the requirement for dealing with the trailing space in the last byte of serialized boolean array.*

Y4_SR_20

The serializer shall pad the last byte of the byte buffer with bits of value 0 while storing the boolean array if the boolean array does not have the number of elements that is divisible by 8.

Y4_SR_21

The `yami_put_boolean_array` function shall place the length of boolean array as an integer value in the given byte buffer, followed by the packed array of boolean values, followed by any trailing zeros if needed for padding.

Note: *Y4_SR_19 specifies the requirement for packing of boolean arrays.*

Note: *Y4_SR_4 specifies the requirement for padding for variable-length sequences.*

Y4_SR_22

The `yami_get_raw_boolean_array` function shall read the specified number of boolean values starting from the beginning of the packed boolean array from the given byte buffer.

Note: *the length prefix is not read, it is assumed that the user has already read the length prefix as an integer value before calling this function.*

Note: *Y4_SR_19 specifies the requirement for packing of boolean arrays.*

Y4_SR_23

The `yami_put_long_long_array` function shall place the length of long long array as an integer value in the given byte buffer, followed by the sequence of long

long array values in the given byte buffer.

Y4_SR_24

The `yami_get_raw_integer_array` function shall read the specified number of integer values starting from the beginning of the integer array from the given byte buffer.

Note: *the length prefix is not read, it is assumed that the user has already read the length prefix as an integer value before calling this function.*

Y4_SR_25

The `yami_put_integer_array` function shall place the length of integer array as an integer value in the given byte buffer, followed by the sequence of integer array values in the given byte buffer.

Y4_SR_26

The `yami_get_raw_long_long_array` function shall read the specified number of long long values starting from the beginning of the long long array from the given byte buffer.

Note: *the length prefix is not read, it is assumed that the user has already read the length prefix as an integer value before calling this function.*

Y4_SR_27

The `yami_put_double_float_array` function shall place the length of double float array as an integer value in the given byte buffer, followed by the sequence of double float array values in the given byte buffer.

Y4_SR_28

The `yami_get_raw_double_float_array` function shall read the specified number of double float values starting from the beginning of the double float array from the given byte buffer.

Note: *the length prefix is not read, it is assumed that the user has already read the length prefix as an integer value before calling this function.*

Y4_SR_29

The `yami_skip_field` function shall skip 4 bytes in the given byte buffer when the requested field type is `yami_boolean` or `yami_integer`.

Y4_SR_30

The `yami_skip_field` function shall skip 8 bytes in the given byte buffer when the requested field type is `yami_long_long` or `yami_double_float`.

Y4_SR_31

The `yami_skip_field` function shall read the integer value from the given byte buffer and then skip the number of bytes in the buffer that is equal to the value read, rounded up to the nearest 4-byte boundary, when the requested field type is `yami_string` or `yami_binary`.

Y4_SR_32

The `yami_skip_field` function shall read the integer value from the given byte buffer and then skip the number of bytes in the buffer that is equal to that length of boolean array after packing and padding if necessary, when the requested field type is `yami_boolean_array`.

Note: *Y4_SR_19 specifies the requirement for packing of boolean arrays.*

Note: *Y4_SR_4 specifies the requirement for padding for variable-length sequences.*

Y4_SR_33

The `yami_skip_field` function shall read the integer value from the given byte buffer and then skip the number of 32-bit words in the buffer that is equal to the value read, when the requested field type is `yami_integer_array`.

Y4_SR_34

The `yami_skip_field` function shall read the integer value from the given byte buffer and then skip the number of 64-bit words in the buffer that is equal to the value read, when the requested field type is `yami_long_long_array` or `yami_double_float_array`.

Y4_SR_35

The `yami_skip_field` function shall read the integer value from the given byte buffer and then repeat, that many times, the complete process of skipping strings or

binary values, when the requested field type is `yami_string_array` or `yami_binary_array`.

Note: *Y4_SR_31 specifies the requirement for skipping strings and binary values.*

Y4_SR_36

The `yami_skip_field` function shall return `yami_unexpected_value` when requested to skip fields of type `yami_nested_parameters` or `yami_nested_parameters_array`.

Note: *skipping of nested fields would be, conceptually, a recursive process.*

Y4_SR_37

The serializer shall format high-level parameters objects in the given byte buffer by placing the length of the object as integer value followed by the given number of fields.

Note: *Y4_SR_38 specifies the requirement for serializing parameters object's fields.*

Y4_SR_38

The serializer shall format the parameters object field in the given byte buffer by placing the field name as a string value, followed by field type, followed by field value, appropriately for its type.

Y4_SR_39

The `yami_fill_message_header` function shall format the high-level message header in the given byte buffer by serializing the parameters object containing 4 fields:

- message type, with the fixed field name "`type`" and fixed field value "`message`",
- object name, with the fixed field name "`object_name`" and the given object name of string type,
- message name, with the fixed field name "`message_name`" and the given message name of string type,
- message identifier, with the fixed field name "`message_id`" and the given message identifier of long long type.

Note: *YR_SR_37 specifies the requirement for serializing parameters objects.*

Y4_SR_40

The `yami_fill_reply_header` function shall format the high-level reply header in the given byte buffer by serializing the parameters object containing 2 fields:

- message type, with the fixed field name "`type`" and fixed field value "`reply`",
- message identifier, with the fixed field name "`message_id`" and the given message identifier of long long type.

Note: *YR_SR_37 specifies the requirement for serializing parameters objects.*

Y4_SR_41

The `yami_fill_exception_header` function shall format the high-level exception header in the given byte buffer by serializing the parameters object containing 3 fields:

- message type, with the fixed field name "`type`" and fixed field value "`exception`",
- message identifier, with the fixed field name "`message_id`" and the given message identifier of long long type,
- exception reason, with the fixed field name "`reason`" and the given reason description of string type.

Note: *YR_SR_37 specifies the requirement for serializing parameters objects.*

Y4_SR_42

The `yami_parse_message_header` function shall parse the given byte buffer, expecting a well-formed parameters object.

Note: *YR_SR_37 specifies the requirement for the format of parameters objects.*

Note: *it is allowed for the parameters object to contain any fields that are well-formed, even if they are not written or read by virtue of other requirements - this allows the message header to have user-defined extensions.*

Y4_SR_43

The `yami_parse_message_header` function shall return the value of string field named "`type`" (for the message type encoded in the parsed header) if such a field is found in the given byte buffer, or an empty string if the field is not present.

Y4_SR_44

The `yami_parse_message_header` function shall return the value of string field named "`object_name`" (for the object name encoded in the parsed header) if such a field is found in the given byte buffer, or an empty string if the field is not present.

Y4_SR_45

The `yami_parse_message_header` function shall return the value of string field named "`message_name`" (for the message name encoded in the parsed header) if such a field is found in the given byte buffer, or an empty string if the field is not present.

Y4_SR_46

The `yami_parse_message_header` function shall return the value of string field named "reason" (for the exception reason encoded in the parsed header) if such a field is found in the given byte buffer, or an empty string if the field is not present.

Y4_SR_47

The `yami_parse_message_header` function shall return the value of long long field named "`message_id`" (for the message identifier encoded in the parsed header) if such a field is found in the given byte buffer, or value 0 if the field is not present.

Y4_SR_48

The `yami_fill_frame_header` function shall format the low-level frame header in the given byte buffer by serializing, in this order:

- low-level message identifier, as an integer value, this value will be non-negative,
- fixed integer value -1,
- high-level message header size, as an integer value, this value will be positive,
- frame payload size, as an integer value, this value will be positive.

Note: *the fixed integer value -1 indicates that the frame is always the first and at the same time the only frame in the whole message.*

Y4_SR_49

The `yami_parse_frame_header` function shall parse the given byte buffer, expecting a well-formed frame header.

Note: *Y4_SR_48 specifies the requirement for the format of frame headers.*

Y4_SR_50

The `yami_parse_frame_header` function shall return the integer value of low-level message identifier, the integer value of high-level message header size and the integer value of frame payload size.

Y4_U_1

The `yami_round_up_to_4` function shall round the given non-negative number up to the nearest value divisible by 4.

Y4_U_2

The `yami_round_up_to_8` function shall round the given non-negative number up to the nearest value divisible by 8.

Y4_U_3

The `yami_strlen` function shall return the number of characters up to the first nul in the given character array.

Note: *this function is a replacement for standard strlen.*

Y4_U_4

The `yami_strcmp` function shall lexicographically (according to character codes) compare two nul-terminated strings and return:

- 0 if the strings are equal,
- negative number of the first string is smaller than the second one,
- positive number if the first string is greater than the second one.

Note: *this function is a replacement for standard strcmp.*

Y4_U_5

The `yami_strncmp` function shall lexicographically (according to character codes) compare two strings of the given length and return:

- 0 if the strings are equal,

- negative number of the first string is smaller than the second one,

- positive number if the first string is greater than the second one.

Note: *this function is a replacement for standard strncmp, except that there is no requirement for not comparing characters following the first encountered nul.*

Y4_U_6

The `yami_strncpy` function shall copy a given number of characters, but no longer than to the first encountered nul, between two strings.

Note: *this function is a replacement for standard strncpy.*

Y4_U_7

The `yami_strfind` functions shall find the given character within a string and return its lowest position or -1 if the given character is not found.

Y4_U_8

The `yami_memcpy` function shall copy the given number of bytes between two byte arrays.

Note: *this function is a replacement for standard memcpy.*

Y4_U_9

The `yami_bzero` function shall clear the array of bytes of the given size by writing zeros to each element of the array.

Note: *this function is a replacement for standard bzero.*

Y4_U_10

The `yami_uint32_to_string` function shall format the given unsigned integer number as a string in the given character buffer.

Y4_U_11

The `yami_uint32_to_string` function shall return the last index of the formatted string or -1 if the given character buffer was too small for successful formatting.

Y4_U_12

The `yami_string_to_uint32` function shall parse and convert the given string to 32-bit unsigned integer number and return 1 if the conversion was successful or 0 if the string is malformed.

Y4_U_13

The `yami_string_to_uint8` function shall parse and convert the given string to 8-bit unsigned integer number and return 1 if the conversion was successful or 0 if the string is malformed.

Y4_NU_1

The `yami_parse_host` function shall parse the given string, expecting a proper host IP address in the "`xxx.xxx.xxx.xxx`" format.

Y4_NU_2

The `yami_parse_host` function shall convert the host IP address to 32-bit unsigned integer in the network byte order.

Y4_NU_3

The `yami_format_target` function shall format, in the given character buffer, the endpoint based on the protocol, host IP address and port number.

Note: *Y4_1 specifies the format for TCP endpoints.*

Note: *Y4_2 specifies the format for UDP endpoints.*

Y4_NU_4

The `yami_parse_endpoint` function shall parse the given string, expecting a

proper endpoint description.

Note: *Y4_1 specifies the format for TCP endpoints.*

Note: *Y4_2 specifies the format for UDP endpoints.*

Y4_NU_5

The `yami_parse_endpoint` function shall extract from the given endpoint description the protocol, host IP address and port.

Note: *Y4_NU_2 specifies the requirement for the format of host IP address value.*

Y4_NU_6

The `yami_parse_endpoint` function shall return yami_bad_protocol if the protocol is not recognized in the endpoint description.

Y4_NU_7

The `yami_parse_endpoint` function shall return yami_unexpected_value if the endpoint format is not recognized.

Y4_NU_8

The `yami_set_reuseaddr` function shall set the SO_REUSEADDR property on the given socket.

Y4_NU_9

The `yami_set_reuseaddr` function shall invoke the optional user-provided error handler if the SO_REUSEADDR property cannot be set.

Y4_NU_10

The `yami_set_nonblocking` function shall set the O_NONBLOCK property on the given socket.

Y4_NU_11

The `yami_set_nonblocking` function shall invoke the optional user-provided error handler if the O_NONBLOCK property cannot be set.

Y4_NU_12

The `yami_set_nodelay` function shall set the TCP_NODELAY property on the given socket.

Y4_NU_13

The `yami_set_nodelay` function shall invoke the optional user-provided error handler if the TCP_NODELAY property cannot be set.

Y4_NU_14

The `yami_set_keepalive` function shall set the SO_KEEPALIVE property on the given socket.

Y4_NU_15

The `yami_set_keepalive` function shall invoke the optional user-provided error handler if the SO_KEEPALIVE property cannot be set.

Y4_NU_16

The `yami_create_tcp_listener` function shall create the listening socket for accepting TCP connections, based on the given host IP address and port number.

Y4_NU_17

The `yami_create_tcp_listener` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_NU_18

The `yami_create_tcp_listener` function shall set the SO_REUSEADDR property on the newly created socket when this is specified in the optionally provided options object.

Y4_NU_19

The `yami_create_tcp_listener` function shall set the backlog property on the newly created socket to the value specified in the optionally provided options object.

Y4_NU_20

The `yami_create_udp_listener` function shall create the socket for accepting incoming UDP packets, based on the given host IP address and port number.

Y4_NU_21

The `yami_create_udp_listener` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_NU_22

The `yami_create_listener` function shall create the listening socket, TCP or UDP, appropriately for the given endpoint name.

Note: *Y4_1 specifies the format for TCP endpoints.*

Note: *Y4_2 specifies the format for UDP endpoints.*

Y4_NU_23

The `yami_clean_listener` function shall clean system resources for the given listener socket.

Y4_NU_24

The `yami_clean_listener` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_NU_25

The `yami_create_outgoing_tcp_conn` function shall create the TCP connection to the given endpoint name.

Y4_NU_26

The `yami_create_outgoing_tcp_conn` function shall set the non-blocking property for the new connection if an appropriate agent option is set.

Note: *Y4_NU_10 specifies the requirement for setting the non-blocking connection property.*

Y4_NU_27

The `yami_create_outgoing_tcp_conn` function shall use the TCP connection timeout option value if the connection is to be created in the non-blocking mode.

Y4_NU_28

The `yami_create_outgoing_tcp_conn` function shall return `yami_timed_out` if the TCP connection is created with in the non-blocking mode with the timeout option and it cannot be established within the given time.

Y4_NU_29

The `yami_create_outgoing_tcp_conn` function shall set the no-delay property for the new connection if an appropriate agent option is set.

Note: *Y4_NU_12 specifies the requirement for setting the no-delay connection property.*

Y4_NU_30

The `yami_create_outgoing_tcp_conn` function shall set the keepalive property for the new connection if an appropriate agent option is set.

Note: *Y4_NU_14 specifies the requirement for setting the keepalive connection property.*

Y4_NU_31

The `yami_create_outgoing_tcp_conn` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_NU_32

The `yami_create_outgoing_udp_conn` function shall create the UDP connection to the given endpoint name.

Y4_NU_33

The `yami_create_outgoing_udp_conn` function shall invoke the optional user-provided error handler and return yami_io_error if its operations cannot be successfully executed.

Y4_NU_34

The `yami_create_outgoing_connection` function shall create the outgoing connection, TCP or UDP, appropriately for the given endpoint name.

Note: *Y4_1 specifies the format for TCP endpoints.*

Note: *Y4_2 specifies the format for UDP endpoints.*

Y4_NU_35

The `yami_accept_tcp` function shall create a new incoming connection from the given listening and ready for processing socket.

Y4_NU_36

The `yami_accept_tcp` function shall set the non-blocking property for the newly accepted connection if an appropriate agent option is set.

Note: *Y4_NU_10 specifies the requirement for setting the non-blocking connection property.*

Y4_NU_37

The `yami_accept_tcp` function shall set the no-delay property for the newly accepted connection if an appropriate agent option is set.

Note: *Y4_NU_12 specifies the requirement for setting the no-delay connection property.*

Y4_NU_38

The `yami_accept_tcp` function shall set the keepalive property for the newly accepted connection if an appropriate agent option is set.

Note: *Y4_NU_14 specifies the requirement for setting the keepalive connection property.*

Y4_NU_39

The `yami_accept_tcp` function shall invoke the optional user-provided error

handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_NU_40

The `yami_clean_connection` function shall clean system resources for the given connection.

Y4_NU_41

The `yami_clean_connection` function shall invoke the optional user-provided error handler and return yami_io_error if its operations cannot be successfully executed.

Y4_NU_42

The `yami_write_tcp` function shall write the given sequence of bytes to the TCP connection.

Y4_NU_43

The `yami_write_tcp` function shall report the number of bytes that were actually written.

Note: *If the connection was configured as non-blocking, it might be possible to write only some of the requested number of bytes.*

Y4_NU_44

The `yami_write_tcp` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_NU_45

The `yami_send_udp` function shall write the given sequence of bytes to the UDP socket.

Y4_NU_46

The `yami_send_udp` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_NU_47

The `yami_read_tcp` function shall read the sequence of bytes that is available in the given TCP socket and place it in the user-provided data buffer.

Y4_NU_48

The `yami_read_tcp` function shall report the number of bytes that were actually read.

Y4_NU_49

The `yami_read_tcp` function shall return `yami_channel_closed` when the connection was physically reset or when the end of file condition was met.

Y4_NU_50

The `yami_read_tcp` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_NU_51

The `yami_receive_udp` function shall read the sequence of bytes that is available in the given UDP socket.

Y4_NU_52

The `yami_receive_udp` function shall report the number of bytes that were actually read.

Y4_NU_53

The `yami_receive_udp` function shall report the remote endpoint name.

Note: *Y4_2 specifies the format for UDP endpoints.*

Y4_NU_54

The `yami_receive_udp` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_NU_55

The `yami_wait_for_work` function shall mark those channels and sockets that are ready for being processed according to their intended use.

Note: *The channel or socket is considered ready if there is data available for reading, buffer space available for writing or incoming connection waiting on the listening socket.*

Note: *The agent logic determines whether, at the time of any given invocation, each particular channel is intended for reading, writing or both; this determination is dynamic and reflects the user-initiated actions as well as the YAMI4 messaging protocol.*

Note: *The channels are marked by setting their respective flags socket_ready_for_input_ and socket_ready_for_output_.*

Y4_NU_56

The `yami_wait_for_work` function shall not take the listening socket under consideration if in the total set of channels there is no place to create any new connection.

Y4_NU_57

The `yami_wait_for_work` function shall return `yami_bad_state` if none of the existing channels is intended for any use.

Note: *The state where no channel is intended for any use is possible when for all channels, the channel's input buffers are full (then the channel is not intended for reading), there is no pending outgoing message waiting for transmission (then the channel is not intended for writing), and there is no place to create a new incoming connection or there is no listener (then the listener is not intended for use). The agent cannot resolve this situation on its own, but the yami_bad_state return value is propagated to the agent's user, who will resolve this by taking appropriate actions that can change any of these blocking conditions.*

Y4_NU_58

The `yami_wait_for_work` function shall return `yami_timed_out` if the timeout value was specified and after the given time none of the channels and sockets was detected as ready for being processed.

Y4_NU_59

The `yami_wait_for_work` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully

executed.

Y4_CHN_1

The `yami_channel_init` function shall initialize the channel object to the state that can be recognized as "not used".

Note: *The "not used" state indicates that a new connection can be created in the slot of the given channel and that it should not participate in waiting for pending I/O operation.*

Y4_CHN_2

The `yami_channel_is_open` function shall determine if the given channel encapsulates an open connection.

Note: *The `yami_channel_is_open` returns false after the channel is initialized as per Y4_CHN_1.*

Y4_CHN_3

The `yami_channel_name_cmp` function shall return true if the given endpoint name matches that of the channel.

Y4_CHN_4

The `yami_channel_open` function shall create a new TCP or UDP channel according to the given endpoint.

Note: *Y4_1 specifies the format for TCP endpoints.*

Note: *Y4_2 specifies the format for UDP endpoints.*

Y4_CHN_5

The `yami_channel_open` function shall return `yami_bad_state` if the given channel is already open.

Y4_CHN_6

The `yami_channel_open` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_CHN_7

The `yami_channel_close` function shall close the given channel and set it to the "not used" state.

Y4_CHN_8

The `yami_channel_close` function shall return `yami_bad_state` if the given channel is already in the "not used" state.

Y4_CHN_9

The `yami_channel_close` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_CHN_10

The `yami_channel_post_raw_message` function shall place the raw (unstructured) message into the output buffer of the given channel.

Y4_CHN_11

The `yami_channel_post_raw_message` function shall return `yami_not_enough_space` if the given message is too big to be placed in the output buffer or if the output buffer already contains a pending message.

Y4_CHN_12

The `yami_complete_msg_preparation` function shall place the message body and a proper frame header into the output buffer of the given channel.

Note: *This function is a common component for functions dealing with messages, replies and rejections, which differ only in their handling of message headers.*

Y4_CHN_13

The `yami_complete_msg_preparation` function shall return `yami_not_enough_space` if the given message body is too big to be placed in the output buffer.

Y4_CHN_14

The `yami_channel_post_message` function shall place the message header into the output buffer of the given channel.

Y4_CHN_15

The `yami_channel_post_message` function shall place the message body and a proper frame header into the output buffer of the given channel.

Y4_CHN_16

The `yami_channel_post_message` function shall return `yami_not_enough_space` if the given message body is too big to be placed in the output buffer or if the output buffer already contains a pending message.

Y4_CHN_17

The `yami_channel_post_reply` function shall place the reply header, body and a proper frame header into the output buffer of the given channel.

Y4_CHN_18

The `yami_channel_post_reply` function shall return `yami_not_enough_space` if the given reply is too big to be placed in the output buffer or if the output buffer already contains a pending message.

Y4_CHN_19

The `yami_channel_post_reject` function shall place the rejection header into the output buffer of the given channel.

Y4_CHN_20

The `yami_channel_post_reject` function shall place an empty rejection body and a proper frame header into the output buffer of the given channel.

Y4_CHN_21

The `yami_channel_post_reject` function shall return `yami_not_enough_space` if the rejection is too big to be placed in the output buffer or if the output buffer already contains a pending message.

Y4_CHN_22

The `yami_channel_write_tcp_data` function shall write the next possible

sequence of bytes from the output buffer to the connection managed by a channel that was open in TCP mode.

Y4_CHN_23

The `yami_channel_write_tcp_data` function shall update the given channel's state to reflect the number of bytes that were written.

Note: *The agent uses this state to handle transmission of messages that might take multiple writes and to decide if the channel's buffer is ready to accept a new outgoing message.*

Y4_CHN_24

The `yami_channel_write_tcp_data` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_CHN_25

The `yami_channel_write_udp_data` function shall write the next possible sequence of bytes from the output buffer to the connection managed by a channel that was open in UDP mode.

Y4_CHN_26

The `yami_channel_write_udp_data` function shall set the given channel's state as ready to accept the next outgoing message if the write was successful.

Note: *The UDP messages are always sent entirely within a single write operation.*

Y4_CHN_27

The `yami_channel_write_udp_data` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_CHN_28

The `yami_channel_write_some_data` function shall write the next possible sequence of bytes from the output buffer of the given channel.

Y4_CHN_29

The `yami_channel_write_some_data` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_CHN_30

The `yami_channel_check_header` function shall parse the initial sequence of bytes in the input buffer of the given channel and determine if the message header was already completely received.

Y4_CHN_31

The `yami_channel_check_header` function shall determine the expected size of the frame header or a complete message.

Note: *Depending on the underlying transport (TCP or UDP), the message might be received completely in a single read operation or in multiple consecutive reads. This function determines how many bytes should be read based on what is available already.*

Y4_CHN_32

The `yami_channel_check_body` function shall determine if the message body was already completely received.

Y4_CHN_33

The `yami_channel_read_tcp_data` function shall read some data to the input buffer from the connection managed by a channel open in TCP mode, according to the expected size of the message.

Y4_CHN_34

The `yami_channel_read_tcp_data` function shall determine if the complete message was already read.

Y4_CHN_35

The `yami_channel_read_tcp_data` function shall return `yami_not_enough_space` if the expected size of message (based on its message header) is bigger than the capacity of the channel's input buffer.

Note: *This can indicate that the remote site attempted to send a message that was too big for the local agent to handle, this is a run-time condition that is handled by the agent by means of forcing the channel to close.*

Y4_CHN_36

The `yami_channel_read_tcp_data` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_CHN_37

The `yami_channel_clean_input_buffer` function shall set the channel's state so that a new incoming message can be received into its input buffer.

Y4_CHN_38

The `yami_channel_check_udp_packet` function shall determine if the received packet is a complete message.

Note: *UDP messages are sent and received entirely within a single I/O operation.*

Y4_CHN_39

The `yami_channel_check_udp_packet` function shall drop the incoming UDP packet if it is determined to be corrupted or incomplete.

Y4_CHN_40

The `yami_channel_read_udp_data` function shall read the UDP packet to the input buffer of the given channel.

Y4_CHN_41

The `yami_channel_read_udp_data` function shall determine if the complete message was read.

Y4_CHN_42

The `yami_channel_read_udp_data` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_CHN_43

The `yami_channel_read_some_data` function shall read the next possible sequence of bytes to the input buffer of the given channel.

Y4_CHN_44

The `yami_channel_read_some_data` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_AGN_1

The `yami_common_init` function shall initialize the agent object to the clean state.

Note: *This function will be called as part of the first invocation on the newly allocated agent object.*

Y4_AGN_2

The `yami_common_init` function shall initialize all channels that belong to the given agent.

Note: *Y4_CHN_1 specified the requirement for initializing channel objects.*

Y4_AGN_3

The `yami_find_free_channel` function shall return the index of first slot that identifies an unused channel.

Y4_AGN_4

The `yami_find_free_channel` function shall return yami_not_enough_space if all channels are already used.

Y4_AGN_5

The `yami_find_channel` function shall return the index of the channel with the given endpoint name.

Y4_AGN_6

The `yami_find_channel` function shall return `yami_no_such_name` if there is

no channel with the given endpoint name.

Y4_AGN_7

The `yami_init` function shall initialize the agent object.

Note: *Y4_AGN_1 specifies the requirement for initializing the agent object.*

Y4_AGN_8

The `yami_init` function shall initialize the agent's options to their default values.

Note: *Y4_OPT_1 specifies the requirement for initializing the options object.*

Y4_AGN_9

The `yami_init_with_user_array` function shall initialize the agent object with the user-provided array of channels for all future operations.

Note: *Y4_AGN_1 specifies the requirement for initializing the agent object.*

Y4_AGN_10

The `yami_init_with_user_array` function shall initialize the agent's options to their default values.

Note: *Y4_OPT_1 specifies the requirement for initializing the options object.*

Y4_AGN_11

The `yami_init_with_options` function shall initialize the agent object.

Note: *Y4_AGN_1 specifies the requirement for initializing the agent object.*

Y4_AGN_12

The `yami_init_with_options` function shall initialize the agent's options to the given values.

Y4_AGN_13

The `yami_init_with_options_ua` function shall initialize the agent object with the user-provided array of channels for all future operations.

Note: *Y4_AGN_1 specifies the requirement for initializing the agent object.*

Y4_AGN_14

The `yami_init_with_options_ua` function shall initialize the agent's options to the given values.

Y4_AGN_15

The `yami_clean` function shall close the listener socket if it was allocated by the given agent.

Y4_AGN_16

The `yami_clean` function shall clean all channels managed by the given agent.

Y4_AGN_17

The `yami_clean` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_AGN_18

The `yami_set_io_error_callback` function shall set the user-provided error handler for further reference.

Note: *The user-provided handler, if provided, can be called in multiple failure scenarios related to I/O or system-related errors.*

Y4_AGN_19

The `yami_set_raw_message_callback` function shall set the user-provided callback for delivering raw (unstructured) messages.

Note: *The user-provided callback is invoked when a new incoming message is completely received by the agent.*

Note: *The raw message callback is independent on the high-level message*

callback; if registered, both will be invoked when the incoming message is completed.

Y4_AGN_20

The `yami_set_message_callback` function shall set the user-provided callback for delivering high-level (structured) messages.

Note: *The user-provided callback is invoked when a new incoming message is completely received by the agent.*

Note: *The high-level message callback is independent on the raw message callback; if registered, both will be invoked when the incoming message is completed.*

Y4_AGN_21

The `yami_set_connection_callback` function shall set the user-provided callback for notifications related to newly created connections.

Note: *The user-provided callback is invoked when a new connection (either outgoing or incoming) is created. This involves also creation of UDP channels, even though they are not physically "connected".*

Y4_AGN_22

The `yami_set_listener` function shall create the TCP or UDP listening socket according to the given local endpoint name.

Note: *Y4_1 specifies the format for TCP endpoints.*

Note: *Y4_2 specifies the format for UDP endpoints.*

Y4_AGN_23

The `yami_set_listener` function shall return `yami_bad_state` if the listener was already created for the given agent.

Y4_AGN_24

The `yami_set_listener` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_AGN_25

The `yami_ensure_connection` function shall create the new outgoing connection if the channel with the given endpoint name does not yet exist.

Y4_AGN_26

The `yami_ensure_connection` function shall return `yami_not_enough_space` if there is no free channel slot to create new outgoing connection and the channel with the given name does not yet exist.

Y4_AGN_27

The `yami_ensure_connection` function shall return the index of newly created channel.

Y4_AGN_28

The `yami_ensure_connection` function shall invoke the connection callback with the given event name if the new connection is created and the connection callback was already set up.

Y4_AGN_29

The `yami_ensure_connection` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_AGN_30

The `yami_open_connection` function shall create the new outgoing connection if the channel with the given endpoint name does not yet exist.

Y4_AGN_31

The `yami_open_connection` function shall return `yami_not_enough_space` if there is no free channel slot to create new outgoing connection and the channel with the given name does not yet exist.

Y4_AGN_32

The `yami_open_connection` function shall invoke the connection callback with the `yami_new_outgoing_connection` event name if the new connection is

created and the connection callback was already set up.

Y4_AGN_33

The `yami_open_connection` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_AGN_34

The `yami_close_connection` function shall clean the resources managed by a channel that has the given endpoint name.

Y4_AGN_35

The `yami_close_connection` function shall invoke the connection callback with the `yami_connection_closed` event name if the connection was found and closed and the connection callback was already set up.

Y4_AGN_36

The `yami_close_connection` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_AGN_37

The `yami_is_output_channel_busy` function shall determine if the channel with the given endpoint name has a pending data in its output buffer.

Note: *The result of this function allows the user to decide if it is safe to post a new outgoing message in the given channel.*

Note: *The channel output buffer can store at most one pending message in its output buffer.*

Y4_AGN_38

The `yami_is_output_channel_busy` function shall return `yami_no_such_name` if there is no channel with the given endpoint name.

Y4_AGN_39

The `yami_post_raw_message` function shall place the raw (unstructured)

message into the output buffer of the channel with the given endpoint name.

Y4_AGN_40

The `yami_post_raw_message` function shall return `yami_not_enough_space` if the given message is too big to be placed in the output buffer or if the output buffer already contains a pending message.

Note: *Y4_AGN_33 specifies the requirement for determining whether the given channel's output buffer can accept new outgoing messages.*

Y4_AGN_41

The `yami_post_raw_message` function shall return `yami_no_such_name` if there is no channel with the given endpoint name.

Y4_AGN_42

The `yami_post_message` function shall place the high-level message into the output buffer of the channel with the given endpoint name.

Y4_AGN_43

The `yami_post_message` function shall return `yami_not_enough_space` if the given message is too big to be placed in the output buffer or if the output buffer already contains a pending message.

Note: *Y4_AGN_33 specifies the requirement for determining whether the given channel's output buffer can accept new outgoing messages.*

Y4_AGN_44

The `yami_post_message` function shall return `yami_no_such_name` if there is no channel with the given endpoint name.

Y4_AGN_45

The `yami_post_reply` function shall place the high-level message into the output buffer of the channel with the given endpoint name.

Y4_AGN_46

The `yami_post_reply` function shall return `yami_not_enough_space` if the

given message is too big to be placed in the output buffer or if the output buffer already contains a pending message.

Note: *Y4_AGN_33 specifies the requirement for determining whether the given channel's output buffer can accept new outgoing messages.*

Y4_AGN_47

The `yami_post_reply` function shall return `yami_no_such_name` if there is no channel with the given endpoint name.

Y4_AGN_48

The `yami_post_reject` function shall place the high-level message into the output buffer of the channel with the given endpoint name.

Y4_AGN_49

The `yami_post_reject` function shall return `yami_not_enough_space` if the given message is too big to be placed in the output buffer or if the output buffer already contains a pending message.

Note: *Y4_AGN_33 specifies the requirement for determining whether the given channel's output buffer can accept new outgoing messages.*

Y4_AGN_50

The `yami_post_reject` function shall return `yami_no_such_name` if there is no channel with the given endpoint name.

Y4_AGN_51

The `yami_dispatch_incoming_message` function shall invoke the user-provided raw message callback with the given channel's input buffer content if such callback was already set up.

Y4_AGN_52

The `yami_dispatch_incoming_message` function shall parse the message header from the given channel's input buffer and invoke the user-provided high-level message callback with `yami_incoming_message` as message type, object name, message name, message identifier and message body if such callback was already set up and the header's message type field indicates that it is a regular

message.

Note: *Y4_SR_39 specifies the requirement for the message header format for regular messages.*

Y4_AGN_53

The `yami_dispatch_incoming_message` function shall parse the message header from the given channel's input buffer and invoke the user-provided high-level message callback with `yami_message_reply` as message type, message identifier and message body if such callback was already set up and the header's message type field indicates that it is a message reply.

Note: *Y4_SR_40 specifies the requirement for the message header format for message replies.*

Y4_AGN_54

The `yami_dispatch_incoming_message` function shall parse the message header from the given channel's input buffer and invoke the user-provided high-level message callback with `yami_message_rejected` as message type, rejection reason and message identifier if such callback was already set up and the header's message type field indicates that it is a rejection (exception) message.

Note: *Y4_SR_41 specifies the requirement for the message header format for rejection (exception) messages.*

Y4_AGN_55

The `yami_dispatch_incoming_message` function shall return `yami_unexpected_value` if the message header from the given channel's input buffer is not recognized.

Y4_AGN_56

The `yami_dispatch_incoming_message` function shall clean the given channel's input buffer if the message was was successfully parsed and dispatched.

Y4_AGN_57

The `yami_accept_incoming_tcp_conn` function shall process the TCP listening socket so that the new incoming connection is created.

Y4_AGN_58

The `yami_accept_incoming_tcp_conn` function shall create a new channel in the free channel slot, based on the accepted incoming connection.

Note: *This function will be called only when some free channel exists.*

Y4_AGN_59

The `yami_accept_incoming_tcp_conn` function shall invoke the connection callback with the `yami_new_incoming_connection` event name if the new connection is created and the connection callback was already set up.

Y4_AGN_60

The `yami_accept_incoming_tcp_conn` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_AGN_61

The `yami_accept_incoming_udp_conn` function shall receive new UDP packet from the UDP socket that was set up as an UDP listener.

Note: *The concept of "listener" is not strict when used with regard to UDP and the received packet might come either from the new remote endpoint or from the endpoint that was already involved in communication.*

Y4_AGN_62

The `yami_accept_incoming_udp_conn` function shall create a new channel for the remote endpoint identified by the received UDP packet, if such a channel does not yet exist.

Y4_AGN_63

The `yami_accept_incoming_udp_conn` function shall invoke the connection callback with the `yami_new_incoming_connection` event name if the new channel is created and the connection callback was already set up.

Y4_AGN_64

The `yami_accept_incoming_udp_conn` function shall place the received packet into the input buffer of the channel identified by the respective remote endpoint name.

Y4_AGN_65

The `yami_accept_incoming_udp_conn` function shall dispatch the incoming packet it if forms a complete and well-formed message.

Y4_AGN_66

The `yami_accept_incoming_udp_conn` function shall clean the given channel's input buffer after processing the packet, independently on whether it formed a proper message.

Y4_AGN_67

The `yami_accept_incoming_udp_conn` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Y4_AGN_68

The `yami_do_some_work` function shall analyse, with the given timeout, the set of I/O resources managed by the given agent and process them accordingly to their type and available actions.

Y4_AGN_69

The `yami_do_some_work` function shall accept the new incoming TCP connection if the TCP listener socket was determined to be ready for accepting new connection and there is a free channel slot to create a new channel.

Y4_AGN_70

The `yami_do_some_work` function shall receive the UDP packet that has arrived to the UDP listener if the UDP listener was determined to have available input data and there is a free channel slot to create a new channel.

Y4_AGN_71

The `yami_do_some_work` function shall, for each channel, read the data to the channel's input buffer if the channel was determined to have available input data.

Y4_AGN_72

The `yami_do_some_work` function shall dispatch the incoming message if for any given channel the newly received data forms a complete message.

Y4_AGN_73

The `yami_do_some_work` function shall, for each channel that was determined to have the capacity to send data, write some amount of data from the channel's output buffer.

Y4_AGN_74

The `yami_do_some_work` function shall close the failing channel, if in that channel the recently executed I/O operation was not successful or the End Of File condition was detected or it was detected that the remote side has closed the connection or the incoming message cannot fit in the channel's input buffer.

Y4_AGN_75

The `yami_do_some_work` function shall invoke the user-provided high-level message callback with `yami_message_abandoned` as message type and message identifier that was provided by user when the outgoing message was posted, if the channel to be closed contains a pending outgoing message and such callback was already set up.

Y4_AGN_76

The `yami_do_some_work` function shall invoke the connection callback with the `yami_connection_closed` event name if the channel is closed and the connection callback was already set up.

Y4_AGN_77

The `yami_do_some_work` function shall invoke the optional user-provided error handler and return `yami_io_error` if its operations cannot be successfully executed.

Note: *It is not an error (and therefore yami_io_error is not returned to the user) if any given channel is closed for reasons other than I/O error.*

Revision history

Revision	Comment
1	Initial revision, refers to the 1.3.1 version of the YAMI4Industry source package.