Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 26, 2012

M. Ertl
November 23, 2011

Protocol for Stage Illumination
draft-mertl-psi-02

Abstract

This memo describes a protocol that builds upon UDP/IP to transport illumination and control data for stage, architectural and other illumination requirements.

It may be understood as a spiritual successor of the traditional DMX (Digital MultipleX) specification series, removing it's limitations and adding flexibility and usability enhancements like auto-discovery and metadata, among other useful features.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 26, 2012.

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

This document contains the specification of the Protocol for Stage Illumination. The intent of this protocol is to be as useful and easy to use as [DMX512-A] while doing away with the severe limitations of DMX in terms of flexibility, structure and speed, and also adding several convenience and management features. The most important benefits are:

- virtually unlimited number of nodes
- automated (re-)discovery of all nodes at any time
- larger and distinct data fields, including metadata
- fully bidirectional communication, including reception of data by the PSI Master
- ability to conglomerate individual PSI Reactors into Groups for improved transfer efficiency, as well as simultaneous unicast addressing as individual PSI Reactors, in any combination
- ability to use a very wide range of readily available, even stock, components as transport media, leading to immediate adaptability to many environments
- may be run alongside other traffic, enabling the use of DHCP, or HTTP configuration in PSI Reactors, over the same network infrastructure
- while not recommended, PSI may be run across a preexisting, general network infrastructure including domestic networks without interfering (within limits of bandwidth requirements)

Comments are solicited and should be addressed to the author.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Description

The PSI operates on top of existing protocols, namely UDP/IP. It is bidirectional and geared towards the typical environment found in

stage lighting systems. The theoretical maximum number of nodes is given by the IN (Identification Number) that is eight octets in size (see Section 3.3). In practice, bandwidth and memory limits are expected to set the actual limit for any given installation. A small number, usually exactly one, of controlling nodes, referred to as "PSI Masters", query, prepare, and send values to a potentially vast number of other nodes, referred to as "Reactors". All messages for Reactors are sent to UDP port 7911; all messages for PSI Masters are sent to UDP port 4919. Two different ports are used so that both personalities can be run on a single node.
Even though a PSI Master may have multiple network interfaces for various reasons like limiting bandwidth requirements, the entirety of Reactors connected to all it’s interfaces is normally available as a whole, and is referred to as "Nexus". A PSI Master implementation may however choose to split the Nexus in any way deemed useful. In addition to data, information about all Reactors is gathered by the PSI Master, and can be presented to the user to aid in management and other tasks. Especially useful for this are the user-defined string and the user-defined numbers, because they may be used to indicate physical whereabouts and other distinguishing properties of any Reactor, in addition to a similar set of data that is vendor-specific.

Because the PSI does not use the underlying networks in unconventional ways, it will coexist peacefully, within bandwidth limits, with any other protocol running over the same networks. This enables, for example, the use of HTTP for configuration interfaces and DHCP for IP address assignment. Because a Nexus does not change behavior much during normal operation, even streaming media may be run alongside PSI.

To achieve a significant degree of flexibility and safety, it does not suffice to assign a channel number through the index of the data inside the data stream, like it is done in DMX ([DMX512-A]) , nor on a per-Reactor basis. While that method has the advantage of lower overhead, it has several disadvantages. One is that each message needs to contain every single channel, at least up to the one actually intended for changing: if the 512th channel is supposed to be changed, all 511 previous channels must be sent as well, even if they are not even assigned to any device.

Additionally, a damaged, missing or misplaced value can not be detected, so that transmission errors may lead to incorrect settings in the receiving devices. Explicit channel numbers, on the other hand, allow for selectively sending data for arbitrary channels and in any order and make sure that a datum received truly is intended for a particular channel, instead of possibly being the result of a transmission error.

Additionally, UDP’s checksumming is automatically used, and any intrinsic error detection and correction methods offered by the network are also taken advantage of. For example, PSI’s expected primary deployment platform, Ethernet, uses a 32 bit checksum that provides a great deal of reliability to received data. The addition of a 16 bit checksum in the extended options of DMX proves the necessity of such reliability. While DMX needs to cope with legacy devices and thus can only make this checksum optional, it always has been mandatory in Ethernet.
In addition to that, PSI binds each channel to a specific Reactor, which is not possible with DMX. This way, the user can at any given time check and control, through the PSI Master, which Reactor contains any given channel, while in DMX this can only be achieved by physically examining every single device. The PSI Master allows the user to spot and resolve conflicts without moving, and without changing settings in Reactors. This is especially convenient because settings like these otherwise necessarily make use of vendor-specific interfaces, requiring the user to first study the device documentation before being able to make the necessary changes. The PSI puts this configuration into the PSI Master, where the user actually is located.

If a device needs to be replaced, the use of DMX requires that the new device is configured to mimic the replaced one. This will lead to problems if the replaced device supported functions the new one doesn’t support, like assignment of non contiguous channel numbers. In such a case the entire controlling sequence, and / or other, completely unrelated devices, would need to be changed resp. reconfigured, resulting in a lengthy and error-prone management effort.
Through explicit assignment of channel numbers, paired with binding of channels to their containing devices, a change like that can be done easily: it only requires the physical replacement of the device in question and reassignment of that device’s channels to the new device. If the PSI Master provides an abstract channel plane that is above the physical devices, then the logical arrangement created on top of the plane never needs to be touched. This means that any logical arrangement may be used, unchanged, on any Nexus, as soon as an appropriate abstract plane has been created through the PSI Master.

While DMX controllers may also support such an abstract plane (called "soft patching"), they can only map logical channels to DMX channels, without real relation to physical devices present. This therefore still leaves the problem of non contiguous channel ranges possibly forcing reconfiguration of many unrelated devices, on top of the need to change the soft patching. So, with DMX the user needs to maintain an updated list of all devices present, plus their configuration, to

be able to configure new devices and to manage the soft patching, whereas the PSI Master automatically maintains this list. The list available from a PSI Master therefore is instantly available and, more importantly, always up to date. PSI creates this list on each restart from the devices actually present, and is able to flag removed devices and list newly added devices, without changing the current configuration. Newly (re-)appearing devices can also be dynamically added to the appropriate list, without changing the currently loaded patching.

The meta datatypes defined by PSI allow for a defined and therefore universal way to gather detailed information about the devices present on any given Nexus. These include vendor and device type, channel counts, and data types as well as user assigned identifiers and comments and detailed status information, allowing for a continuous overview of the entire Nexus.

Another advantage is the availability of larger data types, which are particularly useful for devices like scanners, moving heads and anything requiring finer resolution than the brightness of a lamp does. The development of (non-standard) 16 bit DMX by interpreting two consecutive 8 bit values as single 16 bit value shows that such larger data types are needed in many cases. Should even larger or different data be required, PSI provides additional data types in a well-defined and universal way, allowing for easy use of, for example, 64 bit values.

If a channel cannot handle the full range of possible values offered by the Word type required, it informs the PSI Master of it’s value boundaries so it’s limitations can be honored. A channel MUST tolerate reception of values outside these boundaries, and in response switch to Safe Values and set the appropriate Reactor Status.

Additionally, devices requiring unconventional data types may be employed without reinterpreting stock data types, by using the variable length data types defined by PSI. These may be used for images or similar data, but their actual interpretation completely depends on the device using them.

An alternative to using these opaque data types is the MIME type, because while still containing raw data, it features a description of it’s contents so the receiver is able to compare the received description to it’s expectations.

Each message sent to the PSI Master also contains a crude status indicator, allowing for quick notification about unusual circumstances, possibly automatically triggering more detailed

inquiries and / or other measures by the PSI Master. This is especially useful in large installations, because such a Nexus cannot normally be satisfactorily presented as a whole. Since the PSI Master always possesses complete and up to date information, it is able to actively notify the user of any irregularities in all cases.

2.1. Groups

Groups are used to efficiently handle installations that are composed of many Reactors with few channels each. Sending data to each of them in a separate message may unduly impact the available bandwidth, so it may make sense to handle a number of such Reactors in a single multicast message (see Section 3.4). To do so, an identifier is assigned to all Reactors that the new Group shall contain. This identifier may then be used whenever a message applies to more than one member of the Group. Use of this identifier allows all Reactors that are not part of the specific Group to quickly discard the message without having to process it any further. A message directed at members of a Group contains the usual identifiers for the individual Reactors and may therefore convey totally different data to each member.

All multicast traffic, including discoveries, is sent to a single multicast group, see Section 3.8.1 for details.

The Group facility is OPTIONAL for PSI Masters, but any PSI Master implementing it MUST allow the user to disable it. Reactors MUST implement this facility. Grouping may be handled by any means conceivable, be it manual and / or automatic. In any case, GID assignments may be changed at any time.

2.2. Clusters

Clusters are Groups applying the same data to all members (indicated by the flag PSI_NO_FM_CLUSTER being set). From PSI’s point of view, a Cluster is exactly the same as a Group. However, from the user’s point of view, both are distinct. While Groups serve to more efficiently utilise bandwidth in case of large numbers of Reactors with few channels, Clusters serve to efficiently apply the same data values to multiple Reactors, so these Reactors may have a large number of channels but need to be reasonably similar to each other, especially in terms of channel configuration. Therefore, a PSI Master that implements Clusters should distinguish Clusters and Groups, and apply sanity checks whenever the user adds members to a Cluster. Specifically, Clusters composed of Reactors using different channel configurations or data types should require explicit user confirmation. It may allow this regardless, because the Cluster flag MAY be used to convey general instructions (like Node Options), or to

set only a limited subset of channels that may be applicable to all Reactors in the Cluster. For example, all Reactors in a Cluster may have a channel that is mapped at number 10 and uses 32 bit data. So even though the remaining channels of the clustered Reactors may differ, this single channel may be assigned values using a Cluster message. The PSI Master SHOULD check if there are any matching channels in all clustered Reactors when a new one is added to the Cluster, and inform the user of the result. It MAY offer the user a filter to display only the matching channels.

Also, the CLUSTER flag need not be used in all, even most, messages sent to a Reactor. It is perfectly acceptable to, for example, use this flag only to assign values to a single channel out of many, and to do so only occasionally. Of course, the more this flag is used, the higher transmission efficiency will be.

The Cluster facility is OPTIONAL for PSI Masters, but any PSI Master implementing it MUST allow the user to disable it. Reactors MUST implement this facility. Clustering may be handled by any means conceivable, be it manual and / or automatic. In any case, GID assignments may be changed at any time. Because Clusters rely on Groups, a PSI Master implementing Clusters needs to implement Groups, but not vice-versa.

2.3. Current Values

The Current Values are what is being sent by the PSI Master to a Reactor. They are the result of calculations and / or inputs of any sort. Because they can be any value of the possible value range, they are not suited for continued use in case of loss of control. Instead, Safe Values must be used in that case.

2.4. Safe Values

The Safe Values are specific to each channel. They are designed to be used whenever a channel receives no updates from the PSI Master, and can be used continuously without posing danger for the device or audience.

2.5. Message length

Because fragmenting of messages leads to greater impact of packet loss, it should normally be avoided. In order to keep IP from fragmenting messages, the default maximum message length should be 1400, since 1500 is the normal MTU for Ethernet, allowing some room for different configurations.

A Reactor implementation MAY allow the user to change this maximum,

in response to the actual network performance, but it SHOULD NOT be less than 1400. In fact, the maximum message length of Reactors may be significantly larger by default, because the PSI Master is free to stay well below the maximum, and therefore can adapt, possibly automatically, to network loss rate. On the other hand, the PSI Master's maximum message length may change, and an information message is sent to the Reactors, during normal operation, so that the Reactors do not need to be manually reconfigured to stay below a certain size.

An embedded platform therefore is free to choose an absolute maximum message length, so that it can use static buffers, but advertise less than that if deemed reasonable.

2.6. Operating modes

In addition to the DMX-like operation of continually re-sending all data for all channels regardless of changes in their values (Rapid Change / Redundancy mode), PSI offers a mode where only changed values are sent (Slow Change / ACKed mode). This is done by setting the flag PSI_NO_FM_SENDACK (see Section 3.7.5) in the data messages, and checking the response for consistency with the intended values. Because switching from ACKed mode to Redundancy Mode is done by omission of this flag, both modes may be changed at any time, and even arbitrarily mixed operation, down to individual channels, is possible.

A PSI Master MAY implement both of these modes.

- Redundancy mode

  This mode is very much like the DMX mode of operation: the PSI Master refreshes all channels at a high rate, without caring for the proper arrival of the message or individual data. Data that are lost in transit are automatically compensated for by the high number of redundant data sent, guaranteeing that lost or damaged values do not noticeably affect the outcome.
  Redundancy mode is primarily suited to systems where values are changing rapidly, like stage and disco lighting systems. Due to the high refresh rate, this mode requires significantly more bandwidth than the ACKed mode. However, whenever a large number of all channel's values change rapidly, this bandwidth requirement may very well be lower than in ACKed mode because no replies need to be sent. Additionally, the processing and memory overhead and therefore latency of especially the PSI Master is reduced significantly, because no replies need to be waited for, re-requested and compared.

- ACKed mode

  In this mode, the PSI Master only refreshes a channel whenever it's value changes or other circumstances require it, like a node being restarted. To ensure that the values actually and properly reach the channel, the data messages are decorated with the flag PSI_NO_FM_SENDACK (see Section 3.7.5), so the Reactor sends back a message documenting the execution of the original request. If this reply does not arrive within a certain time, the message is re-sent. This time usually is around 200 ms but SHOULD be changeable by the user to allow for very low bandwidth / high latency links or other preferences. Timeout and retransmission happen until a proper reply is received by the PSI Master, or a new value needs to be set for that channel.
  ACKed mode is primarily suited to systems where values change only infrequently, like architectural or artistic installations.
  Except on startup, due to the low refresh rate it consumes only very little bandwidth, so low bandwidth links may be used.
  Because of the requirement for an acknowledge, the PSI Master needs to retain the outstanding requests, process the results, and be able to re-send them if needed. This means memory and processing overhead that may outweigh the bandwidth reduction in some cases.

2.7. Multiple PSI Masters

While normally only one PSI Master is used, multiple PSI Masters may be desirable in certain situations, so PSI Reactors SHOULD be able to deal with that. Reasons include failover, specialized PSI Masters for different Reactors, or even channels, within one Nexus, or different sources of data available only from different PSI Masters.

Any Reactor SHOULD therefore provide an option via it's interface for handling the conflicts (merging), by allowing the user to select a maximum number of PSI Masters that the Reactor will accept commands and data from. A default of one makes sense, so one PSI Master will be the only one data and commands are accepted from, but there is no upper limit besides the Reactor's capabilities. Internally, the Reactor might maintain a list of PSI Masters that have gone through the Initialization (see Section 3.8.2), and recently have contacted it.
It is not recommended for a Reactor to disable the check whether a PSI Master has gone through the Initialization, meaning that it will accept data and commands in any case. Doing so may create problems due to incorrect configurations, so if it is done, the user SHOULD have an option to enable this check.

Whenever one or more of the original PSI Masters cease to communicate

(time out), then the Reactor will replace them with the next PSI Masters from the list. The way the Reactor picks the PSI Masters (even the initial one) MAY be configurable by the user, and this management may be as sophisticated as desired. A sensible option is to simply go down the list, wrapping around if needed, but it is conceivable to let the user define a fixed selection order, allow only specific, predefined PSI Masters, or allow specific PSI Masters access to different channels of the same Reactor only.
Devices intended to routinely communicate with multiple PSI Masters will likely allow the user to essentially control the entire process, while devices designed primarily for use with one single PSI Master would prefer to not bother the user with unnecessary complexity. In any case, a simple, automatic default configuration SHOULD be preconfigured, so that any Reactor will immediately work, without requiring reconfiguration, in a single PSI Master environment.

If the Reactor is configured to react to more than one PSI Master, conflict resolution methods SHOULD be selectable by the user. A sensible set of choices might include:

- last value persists: always use the most recent value from the PSI Master that sent values most recently.

- first value persists: always use the most recent value from the PSI Master that sent values least recently.

- lowest value persists: compare the most recently sent value of each PSI Master and pick the lowest.

- highest value persists compare the most recently sent value of each PSI Master and pick the highest.

- average of (1 to N) values: the values are stored for each PSI Master so that no two values from the same PSI Master get averaged; N may be lower (but not higher) than the maximum umber of PSI Masters, so that only the most recently received n values from different PSI Masters get averaged.

- free-for-all: allow any PSI Master to do anything, values are replaced by the one most recently received, regardless of the originating PSI Master, as are commands.

Also, a user-configurable timeout is used for when to discard the values sent by one master (timeout), so that even if one PSI Master remains active (sends periodic status inquiries) but does not send values anymore, the stale values can be flushed. Values set using ACKed mode must be kept much longer, so a second timeout value MUST be used for them. The user MUST be able to change both values, and

SHOULD be able to disable this timeout entirely for ACKed mode. The timeouts for switching to Safe Values MAY be used for this purpose. It is advisable that the list provides at least one space more than the maximum number of PSI Masters the Reactor is configured to react to, so that any failing PSI Master can instantly be replaced by another, if present. Otherwise, potential replacement PSI Masters could only go through the Initialization after the failing one has been flagged, unless the Reactor implements some other way of keeping track of excess PSI Masters.

Whenever there are more PSI Masters than spaces in a Reactor's list, the Reactor SHOULD NOT respond to Discoveries sent by the excess PSI Masters. This means that the excess PSI Masters cannot recognize the Reactor at all. If the PSI Master ever gets added into the list, it MUST be responded to normally.

Any PSI Master may have had contact with a Reactor before being dropped from it's list. Any PSI Master querying a Reactor that does currently not respond to it will result in the "PSI_RS_MASTER_IGNORED" status in addition to all other status codes (see Section 3.7.8).

2.7.1. GID Synchronization

A limited amount of synchronization between PSI Masters is required to make the Group and Cluster facility usable by all PSI Masters in a system. If such synchronization is not done, only one PSI Master can manage and GIDs, and the remaining PSI Masters can not use them meaningfully.

To facilitate synchronization, every PSI Master maintains a TCP connection with every other PSI Master. This is acceptable because the number of PSI Masters within a system will normally be very low. Whenever any PSI Master changes GID assignments, it will send a message to every other PSI Master it is connected to (normally, this will be all PSI Masters in the system). Regardless of the command sent to the Reactor, the message sent to the connected PSI Masters contains the full list of GIDs that are assigned to the Reactor, not only the changes. It is of type FM and must only contain one single sentence with the flag PSI_SO_FM_GIDSET set, with the TIN field containing the IN of the Reactor in question. In other words, the entire message is constructed just as it would be sent to a Reactor, but must not be acknowledged by the receiving PSI Master. Since the message is received on the TCP connection from a PSI Master, the receiving PSI Master knows that it receives the message for good reason.

Whenever a PSI Master comes up, every another PSI Master will learn

about it via it's Discovery messages, as it will learn about the other PSI Masters the same way. For every resulting pair of PSI Masters, the PSI Master with the lower IN will initiate the TCP connection to the PSI Master with the higher IN. The rendezvous port number for TCP is the same PSI Master port that is used for UDP.

Whenever a PSI Master comes up, it SHOULD query every Reactor for it's GID assignments, unless it has no intention of making use of the Group or Cluster facilities. This querying can be done as part of the initialization (see Section 3.8.2), or at any later time, but SHOULD be done before the PSI Master starts using or changing GIDs in any way. An exception is if the PSI Master is going to fully redefine all GID assignments, because that can be done without knowledge of any prior assignments.

Ideally, only a single PSI Master should make changes to GID assignments at any given time. Any automatic GID handling SHOULD therefore be disabled by default, so the user will need to selectively enable it on the PSI Master best suited. Manual changes may still be done via any PSI Master and thus cause a race condition if not all PSI Masters are fully synchronized yet, but this will be rare and known to the user, who can make sure that all PSI Masters are up and connected, or will have their lists cleared before connection.

The race condition occurs when more than one PSI Master is set to actively manage GID assignments and any of the following occurs:

a change is in progress while a connection has not yet been established

: a newly started PSI Master might have received the assignment list from a Reactor already, but another, not yet contacted PSI Master changes it's assignments, and the operation completes before contact is made. Thus the newly started PSI Master will never receive the update message and therefore keep the outdated assignment even when it makes contact with the PSI Master that changed this assignment.

the PSI Master has been disconnected but not restarted and changes have been made meanwhile

: a PSI Master will retain it's list during disconnection. Only when the disconnection lasts long enough to make the Reactors consider the PSI Master lost, and subsequently send Discoveries to it when it comes back, will the PSI Master re-read the GIDs. If the duration of the disconnection does not suffice to cause such a timeout, but a GID change has occurred meanwhile, the PSI Master

will maintain the old list.

Automatic conflict detection and correction of the lists can be done by periodically requesting the GID assignment list of every Reactor and comparing that to the stored state, so that synchronization will be re-established after a while. Because any Reactor will be member of only a few groups, the resulting traffic will be low and can be combined with the periodic status inquiries. When to do such re-requests, if at all, depends strongly on the amount of PSI Masters in the Nexus, and also on the frequency of GID assignment changes. In a Nexus with only a single PSI Master, there will be no need to ever do this.
Depending on it’s capabilities, the PSI Master should inform the user about list corrections it had to do, or maybe even prompt for confirmation if the user so wishes.

3. Design

This section describes the construction of messages sent over the medium.

3.1. System-independent representation (transfer syntax)

In order to not add yet another protocol-specific data representation that would require implementors to create yet another, likely non-reusable and probably less tested library, PSI uses the already well-tested CDR (Common Data Representation, [CORBA-CDR]). CDR also forms the basis for CORBA (Common Object Request Broker Architecture) and defines a single format for every primitive type, but two endian representations that can be chosen by the sender. Therefore, every receiver must be prepared to octet-swap if needed. CDR offers data types down to 8 bit, without the overhead of explicit typing and length fields wherever possible. Normally, as defined in the CORBA specification, CDR requires all data types to start at an aligned position matching their own size, with a few exceptions. Since this means overhead of a couple of octets if a large data type follows smaller types that sum up to less than the alignment restriction of the larger type, PSI does not follow this part of the usual CDR practice, and instead uses unaligned CDR.
Also, because CDR does not specify the endianness of bit fields, they are explicitly defined by PSI to match the endianness of the remainder of the message.

Using unaligned CDR does not require creating special libraries because some implementations, like the one of The Ace Orb ([TAO]), already offer the option of turning off the alignment restrictions. Using unaligned CDR, the bandwidth available from a typical, switched 100 Mbit network will suffice for a Nexus larger than two DMX

universes even without use of any optimizations like Groups and Clusters, while maintaining a higher refresh rate even under the worst-case assumption that every single channel has an associated Reactor and thus requires an entire message for a single value. Similarly, 10 Gbit Ethernet can support a Nexus larger than 200 DMX universes, still at a higher refresh rate, over a single cable.

Of course, while PSI has no need for explicit typing done by the transfer syntax, a minimum of meta data still needs to be sent to identify the messages and their contents for flexibility. The resulting hybrid approach is somewhat similar to what is discussed in section 5 (6) of [RFC4506], so PSI does not have the same raw bandwidth efficiency as DMX. The approach taken by PSI is considered by the author to strike a reasonable balance between still allowing for comparatively efficient bandwidth utilization and not sacrificing flexibility to any significant degree.

3.2. Versioning

Even though multiple versions are not planned, and should not be created without proper deliberation, every PSI message, including discovery messages, contains a version field. It MUST be checked by every receiver against it’s list of supported versions to determine how the remainder of the message needs to be interpreted. If a Reactor supports more than one version, it SHOULD respond using the version used by the PSI Master. If the Reactor does not support the version used by the PSI Master, it discards its messages. For every message received in an incompatible version, the Reactor sends a Version Mismatch message (see Section 3.7.4) to the sender. The PSI Master must flag any version differences for the user to see, and do so prominently for incompatibilities.
Likewise, a Reactor that doesn’t use it’s preferred version should, if it provides other indicators to the user, indicate a warning, and an error if it does not have a compatible version to use. A node may support any number of different versions, and allow the user to selectively disable their use, and to specify which is the preferred version.

In order for a node running any protocol version to be able to detect and report version incompatibilities as well as to send Version Mismatch messages to the sender, the message header needs to maintain the structure and contents described in this memo for all versions created. It therefore MUST be modified only in ways that keep the structure in place, like by appending to the basic format. The contents of the basic structure must be made as meaningful as possible.

3.3. Identifier for nodes: IN (Identification Number)

Each node needs to be uniquely identifiable and thus referable within the entire Nexus, for example by a number. The uniqueness of this identifier is very important to guarantee the correct functioning of the installation. To facilitate this without requiring lengthy and error-prone configuration of each Reactor, it needs to even be globally unique. Since PSI is geared towards Ethernet, making use of the MAC-address of the first NIC found in the node makes sense, as it is globally unique already. Regardless, the user may assign the INs in any way convenient as long as they are unique within the Nexus.
Despite the MAC address of Ethernet and most other media types being six octets in length, this field is defined to be 8 octets in length to accommodate the 8 octet MAC format (EUI-64) and to facilitate other uses, like multiple personalities (PSI Master and Reactor).
The MAC octet sequence is filled in in canonical format, with the OUI and the remainder of the MAC separated by as many filler octets with the value 0xFF (and, if necessary, bits with value 1) as needed so that both the OUI and the remainder of the MAC address occupy the outmost octets. This conversion therefore is compatible with the IEEE’s suggested practice, albeit making no distinction between MAC-48 and EUI-48.
Since this number has no corresponding data type, it is defined as sequence of 8 octets. This is no problem in this case, because sequences of octets are stored the same on all architectures, meaning that no endian issues arise from this (see Section 3.1 for details).
Through the IN, identification of nodes does not depend on IP address assignment, so that may be done through DHCP.
In the event that a single node runs more than one personality (PSI Master / Reactor) at the same time, each of them MUST use a different, unique, IN. In a Nexus using the 6 octet MAC format exclusively, this can be accomplished easily by using the two filler octets as an index. Alternatively, and especially when using the 8 octet MAC format, the user may assign artificial INs in any way convenient, provided they are unique within the Nexus.

3.4. Message types

PSI uses a number of basic message types that are further split by direction (to or from the PSI Master). This allows for quick checking whether the received message can be intended for the receiving node at all. For example, a *_FM_* message can not be intended for a PSI Master, not even by other PSI Masters.
Additionally, the meaning of some message types differs depending on direction. Any message may only contain those elements (like sentence types, Node Options, etc.), that apply to it’s direction, because some differ greatly, or even conflict, depending on direction. Therefore it is necessary to check for the proper

direction before using data from a received message.

The various existing and allowed message types are as follows:

- Unicast message

  This is the basic message type and is usable for all nodes. It is sent directly to a single node and therefore creates little processing overhead for any other nodes. Additionally, the entire content of the message is meaningful to the receiving node, so no irrelevant portions require processing. Therefore, messages of this type must only contain information for a single node. However, they may be spread over multiple Node Sections. Use of the flag NODE_FINISHED is optional and does not influence the transmission efficiency. The unicast message uses no GID so this field is omitted in the message header. Even though sending messages of this type as multicast or broadcast can make sense, for example using multiple Node Sections with different TINs, that SHOULD be avoided in favor of the more appropriate multicast message type using a GID to reduce processing overhead for all (or all matching) nodes. See Section 2.1 for details.

  The allowed contents differ depending on direction for this message type, see Constants (Section 5).

- Multicast message

  This message type is intended for use in conjunction with Groups (see Section 2.1), so that the bandwidth can be used more efficiently in installations with many Reactors of only few channels each. A message of this type therefore can contain multiple Node Sections encapsulating contents for different Reactors. As in unicast messages, it is allowed to send multiple Node Sections for the same Reactor in a single message. To increase efficiency, the last node header preceding data for any specific Reactor SHOULD have the flag NODE_FINISHED set, so that the Reactor can discard the remainder of the message immediately. It is recommended that all Node Sections applying to any specific Reactor be consecutive, since that further reduces overhead for all but the last Reactor addressed in the message.

  The allowed contents differ depending on direction for this message type, see Constants (Section 5).

- Discovery message

This is a special message type that must not contain additional data. It's use is restricted to, depending on direction, inform PSI Masters of newly appearing Reactors, or to instruct all Reactors present to send discovery messages, allowing a new or restarted PSI Master to build and maintain an inventory of the Nexus (See Section 3.8 for details). Another use is to directly contact a PSI Master in case it ceased sending messages to a Reactor without apparent reason. This way, the Reactor can initiate being (re-)added to the PSI Master's inventory if, for example, the PSI Master has been restarted but no global discovery request has been made, or in case the Reactor has missed it (it may have been disconnected from the network at that time).
A Reactor receiving a discovery message SHOULD switch to Safe Values; channels using Safety Sequence MUST do so, cease operation, ensure safety if at all possible and also MUST discard any progress through their Safety Sequences.
This message may therefore be sent, depending on the sending node, as unicast or multicast, and is therefore not further subdivided.

- Version Mismatch message

  This is a special message type that must not contain additional data. It's use is to inform the receiver that the sender does not support the protocol version that was used to contact the sender. The version field contains a version supported by the sender. If the sender supports multiple versions, then the first Version Mismatch message contains it's preferred, default version. If the sender receives another incompatible message, then it sends the next version that it supports, until it starts over at the preferred version.
  The receiver of such a message will check if it supports the version indicated, and if it does, then it will establish communication using that version. If it does not, it will collect all versions supported by the sender by sending messages to the sender. The list is complete if the sequence has repeated at least once, possibly more often to allow for duplication, loss and out of order delivery. The receiver may then check this list against it's own support list and pick the version that is nearest to the beginning of both node's lists, or by any other criteria, if multiple matching versions exist. This way, it is ensured that communication is established eventually, and also that, if more than one possible match exists, the version picked is always the same.

  The reason this is done this way is to have as few message types and contents that must be maintained compatibly across all versions as possible. Using Node Specification to list all

supported versions in order of preference, for example, would not only fix that Word type, but also force the entire structure of at least the unicast message to be essentially fixed.

3.5. Headers

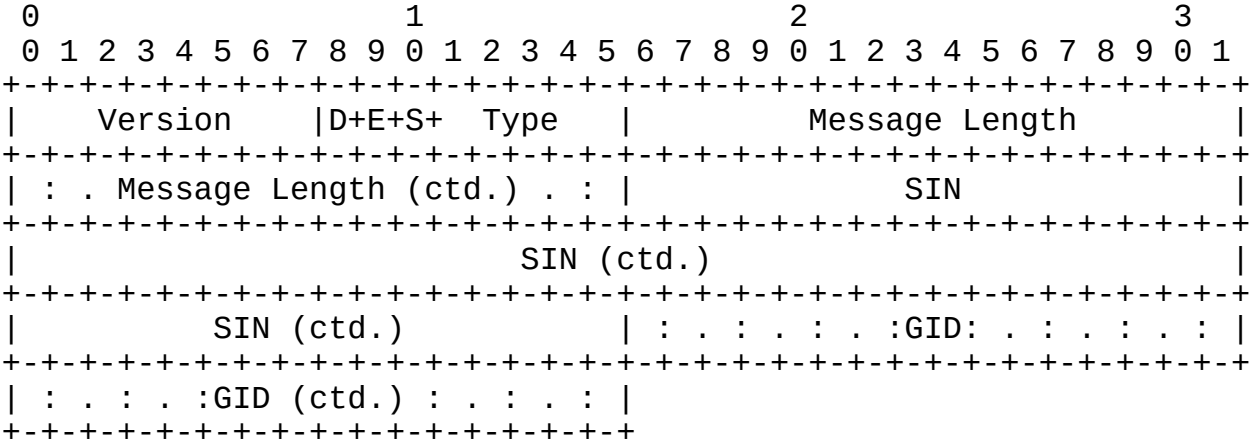
PSI uses three headers that represent different layers of the protocol and are related in a strictly hierarchical manner.

3.5.1. Message header

The message header represents the highest layer of the PSI. Every message must have exactly one message header, as it contains all necessary information like protocol version, source and the type of the message. Additionally, it allows for checking the completeness of the received message in terms of length. The method of specifying the source of the message instead of it’s destination is uncommon and stems from the requirement that the protocol must be able to send information for different Reactors in a single message, in order to reduce the message overhead for Reactors with only minimal data requirements (like those with only a couple of channels). Otherwise messages, and therefore Ethernet frames, would need to be sent for even a single channel. This way, messages for several Reactors can be sent as one single message as multicast (using broadcast is NOT RECOMMENDED), significantly reducing the number of messages that need to be sent (see Section 2.1).

To facilitate this, PSI uses an additional data field in the message header of multicast messages, the Group IDentification (GID). Through use of the GID, no Reactor needs to check every such message received for relevant information. Instead, it may immediately abort processing the message if the GID in the received header does not match any of the GIDs assigned to the Reactor, if there are any. Only messages of type PSI_MT_xx_MULTICAST contain a GID-field. Sending messages of type PSI_MT_xx_UNICAST as multicast or broadcast is NOT RECOMMENDED.

Over the wire format:



Legend:
: . XYZ . : = octets that only exist when certain conditions are met

The PSI message header.

Figure 1

Detailed description of the header fields:

Version: 8 bit

This field contains the version of the sending PSI stack. This is used to determine if the message can be interpreted, and if so, how. See Section 3.2 for details.

Message type: 8 bit

This field specifies the type of the message. The basic types are subdivided into two subtypes each, that define the message direction (from the PSI Master or to the PSI Master); see Section 3.4 for a list and descriptions. These subtypes differ in what their contents may be, and also allow testing for logical correctness. For example, a message sent to the PSI Master cannot be intended for it if it is of subtype "from PSI Master".

Additionally, the endianness and the size of all length fields in the message is defined through bits 1 and 2, respectively, of this field in the following way:

For all message types, bit 1 of the "message type" field indicates the endianness of the entire message (including bit fields): if

bit 1 is set, then the message is in big endian format, otherwise the message is in little endian format. The receiver therefore only needs to convert if the sender uses the other format. An implementation MAY allow the user to choose which endian type to use for sending, regardless of the architecture's natural format. For example, if most Reactors use one format while the PSI Master uses the other, the PSI Master could be configured to use the predominant format for sending, instead of it's native format.

For all message types, bit 2 of the "message type" field indicates the size of all length fields in the entire message: if bit 2 is set, then the length fields are 32 bits in size, otherwise their size is 16 bit.
This is done because in nearly all cases, the maximum message size will be around one Ethernet MTU, or at least well below the 65536 octets possible using 16 bits, so for each length field, two octets less bandwidth are required. Since the message components (Node Sections and Sentences) can only be smaller than the total message length, their length fields will never need to be larger than the one of the message header. In the uncommon case when truly large chunks of data need to be sent in one message, then 32 bits may be used for all length fields. In that case, the additional octets for the length fields will not impact performance due to the large amount of data to be sent.

Thus, including the message direction (bit 0, see Section 3.7.3), the most significant 3 bits are used as flags.

Message length: 16 / 32 bit

This field contains the length of the message in octets, including the message header with the length field itself. The receiver compares it to the number of octets received to see if the message is complete.

SIN: 8 octets

This is the Source Identification Number and contains the sender's IN (see Section 3.3). It informs the receiver of the source of the message. It is used, depending on the message type, for checking (for example, if the sender actually is allowed / supposed to send a particular message type or contents to the receiver) or, especially in case of Discovery messages, to maintain the list of PSI Masters in the Reactors, and the list of Reactors in the PSI Master(s).

GID: 32 bit

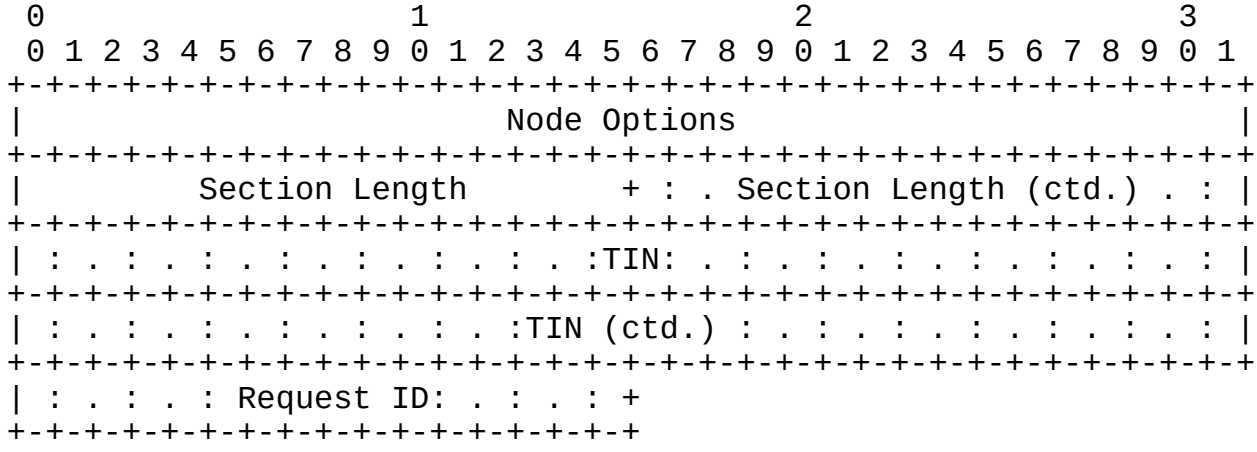
The Group ID is specific to multicast message types. It narrows the scope of the message to possibly only a small subset of the receiving Reactors. By simply comparing the GID to the list of GIDs assigned to itself, a Reactor can decide if it needs to interpret the remainder of the message at all. If a Reactor has no GIDs assigned, it may ignore multicast messages, since they cannot contain information for that Reactor.

Assignment of GIDs is done by the PSI Master and may be changed during normal operation, for example when Reactors vanish or new Reactors appear. The field's size of 32 bit allows for about 4.2 billion unique Groups, which suffices for even large systems.

3.5.2. Node header

The node header is used to identify the target node and marks the beginning of a Node Section. Each message, except for Discovery messages, may contain any number of Node Sections that, depending on the message type, may apply to the same node and / or to different nodes. To facilitate easy jumping across irrelevant (to any particular node) sections, and to allow for checking completeness and correctness, each node header contains the length of the entire Node Section. It also contains the Target Identification Number ("TIN") as well as a field for options that govern the intended use of the data grouped under this header, or represent simple instructions for the entire node. Due to the last part, a node header may be solitary: it does not need to be followed by any Sentence headers (Section 3.5.3).

Over the wire format:



The PSI node header.

Figure 2

Detailed description of the header fields:

Node Options: 32 bit

This is a bit field that contains instructions for the use of the data contained in the current Node Section, as well as self-contained instructions and information. This usually are queries that apply to the entire node, or the flags NODE_FINISHED and REACTOR_ACCEPTED. A description of all Node Options is found in Section 3.7.5.

Section Length: 16 / 32 bit

Similar to the length field of the message header (Section 3.5.1), this field contains the length of the Node Section, including the node header and the length field itself. It allows for checking for completeness and correctness, and easy jumping across parts not interesting to the particular node. Since the next Node Section starts, relative to the start of the length field, at an offset of exactly the amount of octets named in the field, this weeding out can be done relatively easily.

TIN: 8 octets

This is the Target Identification Number and contains the target's IN (Section 3.3). It defines the intended receiver for the current Node Section. If the TIN in any given Node Section does not match the receiver's IN, it must not be interpreted or otherwise used. In this case, a simple jump across the entire Node Section makes sense, using the length field, to not waste CPU time. An implementation MAY also choose to process the Node Section the normal way and just discard the extracted data afterwards, for example if a thorough consistency check is desired. The method chosen has no effect on the data transferred or other nodes.

If the CLUSTER flag is set, the TIN field MUST NOT be present.

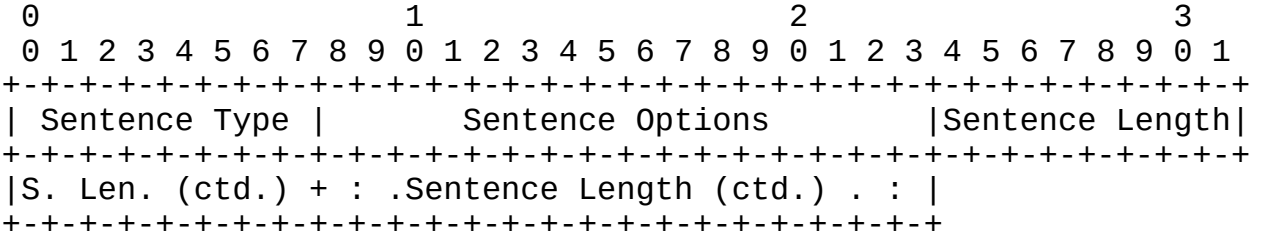
Request ID: 16 bit

This field is present only for requests that have the SENDACK flag set, and for the corresponding acknowledgements (Section 3.7.5). It identifies the request that the acknowledgement refers to, so the receiver can easily match the acknowledgement to the outstanding requests. The sender of a Node Section decorated with the SENDACK flag decides how to generate the number for the next such message, but the method chosen SHOULD ensure that the available number space is fully utilised to minimize chances of collisions in case of duplicated messages.

3.5.3. Sentence header

The Sentence header contains information that applies to the actual data transferred, in order to precisely identify them. It controls which type of Words make up the subordinate sentence. Any given sentence must contain only Words of the same type, but a Node Section may contain any number (including zero) of sentences of any type.

Over the wire format:



Legend:
: . XYZ . : = octets that only exist when certain conditions are met

The PSI sentence header.

Figure 3

Detailed description of the header fields:

Sentence Type: 8 bit

The sentence type specifies the type of the Words making up the current sentence. It is the same for all Words of any given sentence and can thus be used, in combination with the sentence length field, to compute the Word count. This saves an additional field for the Word count. See Section 3.6 for a description of all Word types.

Sentence Options: 16 bit

This field is, like the Node Options field, a bit field, that specifies the data contained in the current sentence as well giving instructions to their use. It is especially used to govern the use of generic Word types like Node Specification and Channel Specification. Additionally, it may contain requests or instructions that are to be applied to the data contained in the sentence. This way, the PSI Master can use a single sentence of type Channel Number to request all information about the channels specified, as well as switch the channels between Safe Values and Current Values. By using Words of one of the Data Word types, new values may be assigned to the specified channels at the same time. In messages to the PSI Master, this field merely specifies the use of the generic Word types.

Sentence Length: 16 / 32 bit

Like the length fields in the message- and node headers, this field contains the number of octets used by the current sentence, including the Sentence header and it's length field. Contrary to the lengths of Node Sections and Messages, it is not meant to facilitate jumping across the sentence, since the relevance of any particular sentence has already been established by the previous examination of the preceding node header. Instead, this field describes the number of Words contained in the sentence. To be consistent with the other length fields and to allow for easy checking of consistency, it contains the total octet count. If the length of the current Word type, which is given in the sentence type, is fixed, then the number of Words contained can be derived through simple division, allowing further check for consistency of Word type and sentence length. If the length of the current Word type is variable, like for example the Reactor Status, the sentence may contain only exactly one Word, with the sentence length representing the actual length. This restriction stems from the deliberation that Words of variable length will be used only occasionally, with their length possibly becoming comparatively large. Therefore, the overhead of a complete Sentence header per Word is acceptable in this case. Since the majority of Words transferred is of fixed length however, the introduction of an additional length field for each Word would cause comparatively large overhead, while it's usefulness would be restricted to comparatively few and seldom used Words and thus marginal. Especially with small Words, like channel numbers, the overhead would be 100% of the actual data and therefore not justified.

3.6. Word types

To further explain the structure of PSI messages, this section lists and describes the types of Words used. The sequence described is the sequence as seen on the network. The ordering of octets and other issues are governed by the transfer syntax (Section 3.1) and therefore not explicated here. Channel number sizes are given by the appropriate Sentence Option. The Word type used, in conjunction with the data boundaries, defines the resolution to use, so the PSI Master knows which range of values is allowed for the channel and can scale the values appropriately. Values within the Word type but outside the defined boundaries MUST make the channel switch to Safe Values and setting of the appropriate Reactor Status.

While a PSI Master MUST implement all of these types except where mentioned otherwise, a Reactor only needs to implement the types of PSI_Word_Datum that it actually uses.

Boolean Data Word: PSI_Word_DatumB

This data Word consists of the channel number and the actual, boolean, datum.
The syllables are sent in the sequence: "channel number, datum".

Signed 8 bit Data Word: PSI_Word_DatumS8

This data Word consists of the channel number and the actual, signed, datum, with a width of 8 bits.
The syllables are sent in the sequence: "channel number, datum".

Unsigned 8 bit Data Word: PSI_Word_DatumU8

This data Word consists of the channel number and the actual, unsigned, datum, with a width of 8 bits.
The syllables are sent in the sequence: "channel number, datum".

Signed 16 bit Data Word: PSI_Word_DatumS16

This data Word consists of the channel number and the actual, signed, datum, with a width of 16 bits.
The syllables are sent in the sequence: "channel number, datum".

Unsigned 16 bit Data Word: PSI_Word_DatumU16

This data Word consists of the channel number and the actual, unsigned, datum, with a width of 16 bits.
The syllables are sent in the sequence: "channel number, datum".

Signed 32 bit Data Word: PSI_Word_DatumS32

This data Word consists of the channel number and the actual, signed, datum, with a width of 32 bits.
The syllables are sent in the sequence: "channel number, datum".

Unsigned 32 bit Data Word: PSI_Word_DatumU32

This data Word consists of the channel number and the actual, unsigned, datum, with a width of 32 bits.
The syllables are sent in the sequence: "channel number, datum".

Signed 64 bit Data Word: PSI_Word_DatumS64

This data Word consists of the channel number and the actual, signed, datum, with a width of 64 bits.
The syllables are sent in the sequence: "channel number, datum".

Unsigned 64 bit Data Word: PSI_Word_DatumU64

This data Word consists of the channel number and the actual, unsigned, datum, with a width of 64 bits.
The syllables are sent in the sequence: "channel number, datum".

Floating-point Data Word: PSI_Word_DatumF32

This data Word supplies 32 bit for a floating-point datum.
The syllables are sent in the sequence: "channel number, datum".

Whenever possible, one of the fixed-point Word types should be used instead of the floating-point representations, so that less complex hardware is needed and interoperability is improved.

Implementation of this type is recommended for PSI Masters, and optional for Reactors.

Double floating-point Data Word: PSI_Word_DatumF64

This data Word supplies 64 bit for a floating-point datum.
The syllables are sent in the sequence: "channel number, datum".

Whenever possible, one of the fixed-point Word types should be used instead of the floating-point representations, so that less complex hardware is needed and interoperability is improved.

Implementation of this type is recommended for PSI Masters, and optional for Reactors.

Quadruple floating-point Data Word: PSI_Word_DatumF128

This data Word supplies 128 bit for a floating-point datum.
The syllables are sent in the sequence: "channel number, datum".

Whenever possible, one of the fixed-point Word types should be used instead of the floating-point representations, so that less complex hardware is needed and interoperability is improved.

Implementation of this type is recommended for PSI Masters, and optional for Reactors.

Fixed-point Data Word: PSI_Word_Datum_Fixedpoint_D16

This data Word provides a fixed-point data representation. An unsigned 64 bit integer is used for the integer part, and a signed 16 bit integer for the divisor. This way, all convenient fixed-point formats can be expressed using a single Word type, so that all implementations supporting this format can interoperate. Through placing the sign bit in the divisor, the integer part can utilise the entire 64 bit range, at the expense of a little complexity on the sending side. The divisor only requires a much smaller range, so halving it has little impact.
The syllables are sent in the sequence: "channel number, integer part, divisor".

This data type should be preferred over the floating-point representations whenever possible, so that less complex hardware is needed and interoperability is improved.

Fixed-point Data Word: PSI_Word_Datum_Fixedpoint_D64

This data Word provides a fixed-point data representation. An unsigned 64 bit integer is used for the integer part, and a signed 64 bit integer for the divisor. This way, all convenient fixed-point formats can be expressed using a single Word type, so that all implementations supporting this format can interoperate. Through placing the sign bit in the divisor, the integer part can utilise the entire 64 bit range, at the expense of a little complexity on the sending side. The divisor only requires a much smaller range, so halving it has little impact.
The syllables are sent in the sequence: "channel number, integer part, divisor".

This data type should be preferred over the floating-point representations whenever possible, so that less complex hardware is needed and interoperability is improved.

Data Word of variable length: PSI_Word_Data_Opaque

This Word allows the transfer of unspecified data. It may be used for exotic systems that cannot be reasonably covered using the other Word types. It is necessary to note that no processing of the supplied data is performed, so that device-dependent issues

are left for the communicating nodes to solve themselves. Merely the channel number is transferred as usual.
Whether or not the querying or using the range for this type makes sense also depends on the communicating nodes.
Because the interpretation depends entirely on the receiver, these generic types are just asking for trouble and should be used only when truly necessary.

This is one of the Word types that have no fixed length and therefore may only occur once per sentence, requiring a sentence per Word. The reason is that the sentence length is used to convey the number of octets making up the Word, which therefore represent the syllables of this type of sentence. See also Section 3.5.3. The syllables are sent in the sequence: "channel number, data octets".

Implementation of this type is completely optional.

Data Word for device-specific data: PSI_Word_Opaque_Plain

This Word differs from PSI_Word_Data_Opaque only in that it does not contain a channel number.
The syllables are sent in the sequence: "data octets".

Implementation of this type is completely optional.

Data Word for text: PSI_Word_Data_Text

This is another Word without fixed length, and is thus only allowed to occur once per sentence because it’s length is given by the sentence length. It’s syllables are single octets that contain purely text data. It is converted appropriately if needed by any specific system architecture. In addition to the data and termination octets, the transfer syntax prepends the actual number of octets, creating an overhead of four octets that also needs to be taken into account in the sentence length.
The syllables are sent in the sequence: "channel number, Word length (implicit), data octets".

The encoding used is UTF-8 ([RFC3629]) (the sender MAY use plain US-ASCII), and it SHOULD be encoded using the string type.

Implementation of this type is completely optional.

Data Word for plain text: PSI_Word_Text_Plain

This Word differs from PSI_Word_Data_Text only in that it does not contain a channel number.
The syllables are sent in the sequence: "Word length (implicit), data octets".

Data Word for MIME content: PSI_Word_Data_MIME

This is another Word without fixed length, and is thus only allowed to occur once per sentence because it’s length is given by the sentence length. It’s data syllables are single octets that contain binary data that are not converted in any way. For the MIME Content-Type syllable itself that uses the same encoding as the String and Text words do, the transfer syntax prepends the actual number of octets, creating an overhead of four octets that also needs to be taken into account in the sentence length.
Being capable of binary transport, no conversion of the MIME data from it’s native representation is allowed. The Content-Type is used according to [RFC2045] (section 5).
The syllables are sent in the sequence: "channel number, MIME Content-Type length (implicit), MIME Content-Type text, data octets".

Implementation of this type is completely optional.

Data Word for plain MIME content: PSI_Word_MIME_Plain

This Word differs from PSI_Word_Data_MIME only in that it does not contain a channel number.
The syllables are sent in the sequence: "Word length (implicit), data octets".

This type is part of the replies to the USI and VSI queries, and can be especially useful when used for user- or vendor defined images that may help with identification of a Reactor. However, implementation of this type MAY be restricted to the capability of sending an empty sentence with a length of zero in case of a Reactor, and to discarding any sentences of this type in case of a PSI Master. Also, though any MIME type MAY be contained in these sentences, it is RECOMMENDED to stick to PNG for images and to vorbis for audio, in order to enhance chances that any given PSI Master will be able to use the contents.
A Reactor MAY use different images for different maximum message lengths, so if a PSI Master supports a greater maximum message length, a correspondingly larger (in terms of octets) image can be

sent.

Data Word for time content: PSI_Word_Data_Time

This type is used for defining current, start, stop, etc. times. The exact use depends on the application, as it may be used, for example, for synchronization and / or for setting up times to start or stop actions, as well as define repetitions and / or intervals. Use for clock synchronization will be affected by network delay and is therefore non-deterministic, so the use of NTP or similar is recommended for that purpose. Another use is the synchronization of data, usually contained in the same message, to some definition of time. This may be media frames, for example, or a channel may be instructed to change the output to the values defined in the message at the specified time.

The syllables are signed 16 bit integers for years and days, and signed 8 bit integers for hours, minutes, seconds and hundredth of seconds, as well as doublets of one unsigned 16 bit integer (integer part) and one signed 16 bit integer (divisor) for the frame rate and frames time, one unsigned 8 bit integer as a flag field and one unsigned 64 bit integer for the repetition count. The syllables are sent in the sequence: "channel number, years, days, hours, minutes, seconds, hundredth of seconds, frame rate integer part, frame rate divisor, frames time integer part, frames time divisor, flags, repetition count".
Any implementation SHOULD be able to cope with the full binary range for the date fields (years, days, etc.), even though it is DISCOURAGED to exceed the conventional limits. Being application specific anyway, handling of leap seconds, etc. is left to the implementation.

The flag field is defined as:

* Bit 7 to indicate the application of the "Drop Frame" method when set

* Bit 6 to indicate the time is relative to now (actually, the arrival / processing of the message, which will be affected by assorted non-deterministic delays) when set

* Bit 5 to indicate it is part of an interval (normally, both start and stop times will be sent consecutively) when set

* Bit 4 to indicate if it is a start (set) or stop (not set) time (usually as part of an interval)

Note: the frame rate and frames time fields need to be able to express fractions, because some video formats like NTSC use non-integer values. To avoid requiring floating-point capabilities, a format similar to the fixed-point data types is used, but with less range due to the smaller numbers commonly used. While using a type list would remove the need for these fields, that would severely reduce flexibility and adaptability. Usually, the occurrence of this Word will be comparatively low or the remaining message content sufficiently long to outweigh this bloat.

Whether or not querying or using the range for this type makes sense depends on the communicating nodes. It might be used to query the frame rate and Drop Frame usage of the Reactor, for example.

Implementation of this type is completely optional.

Data Word for plain time content: PSI_Word_Time_Plain

This Word differs from PSI_Word_Data_Time only in that it does not contain a channel number.
The syllables are sent in the sequence: "years, days, hours, minutes, seconds, hundredth of seconds, frame rate integer part, frame rate divisor, frames time integer part, frames time divisor, flags, repetition count".

Implementation of this type is completely optional.

Data Word for plain Channel Number: PSI_Word_Channel_Number

This Word contains only one syllable, representing a channel number. It is used in conjunction with Sentence Options to send instructions to and to query information about the channel(s) named.
The syllables are sent in the sequence: "channel number".

Generic Data Word for numbers: PSI_Word_Generic_32

This Word also has no fixed length, and is given by the sentence length. It is used for transferring numbers like the user-assigned numeric IDs. All syllables are 32 bits wide and thus the sentence length always is a multiple of four octets. Therefore it can be used to calculate the number of syllables and no additional field is needed to store the actual length.
The syllables are sent in the sequence: "datum, datum,...".

Data Word for Reactor Status: PSI_Word_Reactor_Status

This Word also has no fixed length, and is given by the sentence length. It is used for transferring detailed status information about a Reactor, one per syllable. All syllables are 8 bits wide and thus the sentence length can be used to calculate the number of syllables and no additional field is needed to store the actual length.
The syllables are sent in the sequence: "datum, datum,...".

Data Word for channel status: PSI_Word_Channel_Status

This Word contains two syllables: a channel number and the channel status, that is encoded as bit field of 16 bits.
The syllables are sent in the sequence: "channel number, channel status".

Data Word for Safety Sequence: PSI_Word_Safety_Sequence

This Word is used to convey the sequence required to change the value of a channel that requires extra safety. It has a variable length and thus must only occur once per sentence. It’s component syllables are 32 bits in size and thus the sentence length can be used to calculate the number of syllables, without the need for an additional length field. The first syllable is the channel number; all following syllables are the numbers to send, in the same sequence as given in this Word, to activate the channel.
The syllables are sent in the sequence: "channel number, sequence component 0, sequence component 1,...".

Any such channel MUST NOT change it’s value if the sequence is not received fully, out of order, or numbers not in the sequence are received. To change the value of such a channel, the PSI Master MUST split the sequence into individual messages containing Words of this type, but only a single component of the sequence at a time. The sentence length in this direction therefore MUST be 1. The value that the channel is supposed to switch to MUST be sent in a separate sentence of the same message, with the Word type matching the channel’s data type, with every component of the sequence. All messages that contain a part of the sequence MUST contain the same target value. If the target value in one of the messages does not match the previous, then, even if the sequence component itself matches the one expected, any sequence progress MUST be discarded and the value MUST NOT be changed.

Without the requirement for splitting, chances of transmission

errors creating the valid sequence would still be very low, but because all packets would always contain the correct sequence, a single duplicated packet could trigger a second value change. Unless the sequence length is 1, which is allowed but strongly discouraged, this cannot occur due to the splitting.
However, multiple channels, including channels using Safety Sequence, may be served in a single message.
Because any packet may get lost or received incorrectly, to ensure that the sequence has been advanced, the flag PSI_NO_FM_SENDACK may be used.

Additionally, a channel using Safety Sequence MUST NOT advance the sequence when it is using Safe Values, like after receiving PSI_SO_FM_GOSAFE or PSI_NO_FM_GOSAFE.

The actual length of the sequence is left to be assigned by the users of the system, governed to their safety requirements, so an implementation MUST NOT expect any specific length. A device MAY include a default sequence deemed reasonable by the vendor, but it MUST allow the user to change the sequence, including it's length. A length of 1 MUST NOT be used as default, and an implementation MUST warn the user about the safety implications of setting it to that length. A length of zero MUST make the channel ignore any value changes it may receive, effectively disabling it. A sequence MUST NOT contain any single number more than once, so that duplicated or reordered packets can not advance the sequence. Whenever the sequence received does not match the required sequence, the receiver MUST discard all parts received already, and restart at the beginning. The numbers 0 (all bits cleared) and 0xFFFFFFFF (all bits set) MUST reset the receiver, and therefore cannot be used as part of a sequence. This is done so that any channel using Safety Sequence can safely be reset without knowing the sequence that has been assigned to it.

Note: it would be possible to add explicit numbering to each part of the sequence, but because the sequence length will always be small compared to the available number space, this seems to be unnecessary overhead, adding to the overhead already created by splitting.
A preliminary suggestion is for a channel implementation to have room for a length of at least 20 per channel. Any PSI Master implementing Safety Sequence is expected to allow for longer sequences.

Implementation of this type is completely optional.

Data Word for channel counts: PSI_Word_Channel_Count

This Word is composed of three syllables, each 32 bit in size, specifying the counts of the various channel types in a Reactor. The syllables are sent in the sequence: "input channel count, in/out channel count, output channel count".

Generic Word for Node Specification: PSI_Word_Node_Specification

This Word only contains a single value of 32 bits. It's meaning is defined by the Sentence Options. This method was chosen to not have to define a large number of Words that differ in name only. The syllables are sent in the sequence: "node specification".

Generic Word for Channel specification: PSI_Word_Channel_Specification

This Word contains two syllables: the channel number and a channel specification of size 32 bit. Like PSI_Word_Node_Specification, the actual meaning is given by the Sentence Options.
The syllables are sent in the sequence: "channel number, channel specification".

3.7. Constants

This section lists and explains the constants used in PSI. Since only the values of the constants are transferred between nodes, an implementor may choose to use any convenient naming and assignment scheme.

3.7.1. Conventions

The conventions used in this document are as follows:

- Naming

  All constants are prefixed with "PSI" to emphasize their belonging to the PSI, and to avoid collision and confusion with other constants. An underscore ("_") separates the specification of the general usage in the form "AB". Following another underscore is the direction in form "CD", if the constant is defined for more than one direction. A further underscore separates this from the general meaning of the constant, usually given by an acronym "E". Thus, constant names are constructed like this:

PSI_AB_CD_E (directional constant), like PSI_ST_TM_WD32
PSI_AB_E (nondirectional constant), like PSI_CR_8

Values for AB:

PV: Protocol Version
MT: Message Type
NO: Node Option
SO: Sentence Option
ST: Sentence Type
RT: Reactor Type
RS: Reactor Status
CT: Channel Type
CS: Channel Status

Values for CD:

FM: From PSI Master
TM: To PSI Master

Constants that have different application (like single or conglomerate constants) are named in a way to show their meaning, and are prefixed with "PSI".

- Values

  Especially bit constants are given in the form "bit X", specifying the bit that is set, counting the MSB of the field as bit 0. Other constants, especially large ones, are given as hexadecimal, prefixed by the conventional "0x".

3.7.2. Protocol version

PSI 32 Bit: PSI_PV_0

This is the protocol described here.
It is defined as 0.

3.7.3. Bases for constants

To ease readability and structure, there are a few constants serving as base for others. The bases are chosen in a way to allow the constants of every area to start from 0 by adding the base as offset.

Type base for ’direction from PSI Master’: PSI_TYPE_BASE_FM

This is the base for all constants that depend on the message direction, identifying data coming from the PSI Master.
It’s value is 0.

Type base for ’direction to PSI Master’: PSI_TYPE_BASE_TM

This is the base for all constants that depend on the message direction, identifying data sent to the PSI Master.
It’s value is "bit 7", dividing the available number space by half.

Type base for ’message from the PSI Master’: PSI_MT_BASE_FM

This is the base for all message types that are sent by the PSI Master.
It’s value is PSI_TYPE_BASE_FM.

Type base for ’message from the PSI Master’: PSI_MT_BASE_TM

This is the base for all message types that are sent to the PSI Master.
It’s value is PSI_TYPE_BASE_TM.

Type base for sentences in messages coming from the PSI Master: PSI_ST_BASE_FM

This is the base for sentence types that are used in messages sent by the PSI Master.
It’s value is PSI_TYPE_BASE_FM.

Type base for sentences in messages sent to the PSI Master: PSI_ST_BASE_TM

This is the base for sentence types that are used in messages sent to the PSI Master.
It’s value is PSI_TYPE_BASE_TM.

3.7.4. Message types

The following constants identify the message types as described in Section 3.4. The direction of the message is identified by use of the appropriate constant. The meaning of a message may differ depending on it's direction, so this is not only important for plausibility checking.

As described in Section 3.5.1, including the message direction (bit 0), the most significant 3 bits are used as flags.

Messages from the PSI Master:

Normal (unicast) message: PSI_MT_FM_NORMAL

This designates the most basic type of data message. The message is directed at a single Reactor only and therefore does not contain a field for a GID. It may, however, contain multiple Node Sections.
It's value is (PSI_MT_BASE_FM+0).

Multicast message: PSI_MT_FM_MULTICAST

This type designates a message intended for multicasting. Contrary to the unicast message, it contains a field for a GID.
It's value is (PSI_MT_BASE_FM+1).

Discovery message: PSI_MT_FM_DISCOVERY

This type designates a PSI Master to Reactor Discovery message. It does not contain information besides the message header itself.
It's value is (PSI_MT_BASE_FM+2).

Version mismatch message: PSI_MT_FM_VERSION_MISMATCH

This type designates a PSI Master to Reactor version mismatch message. It does not contain information besides the message header itself. The version field indicates the next supported version.
It's value is (PSI_MT_BASE_FM+3).

Messages to the PSI Master:

Normal (unicast) message: PSI_MT_TM_NORMAL

This message is directed at a single PSI Master. This is the usual case even in the presence of multiple PSI Masters. It's value is (PSI_MT_BASE_TM+0).

Multicast message: PSI_MT_TM_MULTICAST

This type designates a message intended for multicasting. Contrary to the unicast message, it contains a field for a GID. It's value is (PSI_MT_BASE_TM+1).

Discovery message: PSI_MT_TM_DISCOVERY

This type designates a Reactor to PSI Master Discovery message. It does not contain information besides the message header itself. It's value is (PSI_MT_BASE_TM+2).

Version mismatch message: PSI_MT_TM_VERSION_MISMATCH

This type designates a Reactor to PSI Master version mismatch message. It does not contain information besides the message header itself. The version field indicates the next supported version.
It's value is (PSI_MT_BASE_FM+3).

3.7.5. Node Options

This section describes the constants defined for Node Options. The Node Options are encoded as bit field to conserve bandwidth.

Node Options for messages from the PSI Master:

These Node Options are valid coming from the PSI Master. They may be combined in any number, with certain exceptions. This way it is possible to request all possible information about a Reactor with a single message. Some requests may result in more data to be sent than the current maximum message length. Unless a single Word exceeds the maximum message length (which is an error; truncation is only possible for non-essential things like vendor-specific information), the reply is split across multiple messages, using proper sentences.

Query all data types: PSI_NO_FM_DTREQ

This flag is used to request transmission of the data types of all channels to the PSI Master. If the resulting list is longer than the maximum message length, it may be split into separate messages.
It’s value is "bit 31".

Query all channel types: PSI_NO_FM_CTREQ

This flag is used to request transmission of the types of all channels to the PSI Master. If the resulting list is longer than the maximum message length, it may be split into separate messages.
It’s value is "bit 30".

Query all channel’s data boundaries: PSI_NO_FM_DBREQ

This flag is used to request transmission of the boundaries of all channels to the PSI Master. If the resulting list is longer than the maximum message length, it may be split into separate messages.
It’s value is "bit 29".

Query all channel states: PSI_NO_FM_CSREQ

This flag is used to request transmission of the status of all channels to the PSI Master. If the resulting list is longer than the maximum message length, it may be split into separate messages.
It’s value is "bit 28".

Query all data values: PSI_NO_FM_VREQ

This flag is used to request transmission of the Current Values of all channels to the PSI Master. If the resulting list is longer than the maximum message length, it may be split into separate messages.
It’s value is "bit 27".

Query the Reactor’s status: PSI_NO_FM_RSREQ

This flag makes the Reactor send a detailed status message to the PSI Master.
It’s value is "bit 26".

Query maximum message length: PSI_NO_FM_MMLREQ

This flag is used to request transmission of the Reactor’s maximum message length to the PSI Master. This SHOULD be at least around 1400 octets, but MAY be significantly larger. The PSI Master MUST NOT send messages larger than this to the Reactor (including Group messages, so these will always be at most the size of the lowest maximum of all Reactors in the group), but may send any size below this. If a Reactor’s maximum message size changes, it MUST set the PSI_RS_REREAD_ALL status.
Note that even though the Reactor must know the PSI Master’s maximum message length, there is no corresponding request flag. That is because the PSI Master will send this information by itself, usually alongside it’s REACTOR_ACCEPTED message. If for some reason the Reactor has not received this information despite a REACTOR_ACCEPTED message, and is sent other requests or data, it MUST resume responding to Discovery messages, discarding the REACTOR_ACCEPTED message.
It’s value is "bit 25".

Query maximum refresh rate: PSI_NO_FM_MRRREQ

This flag is used to request transmission of the Reactor’s maximum refresh rate to the PSI Master. This MUST be at least 100 Hz, but MAY be significantly larger. The Reactor MUST be able to sustain this rate indefinitely, under all circumstances that may arise during normal operation, especially in the face of any combination of group, cluster and other messages, and regardless of how the channel data are distributed across the messages.
Channels using Safety Sequence may depend on the length of the sequence, so a separate refresh rate is sent for these, based on the current sequence’s length.
The PSI Master SHOULD NOT refresh any Reactor faster than it’s advertised maximum refresh rate, so this may affect grouping and clustering decisions.
The PSI Master MAY decide to refresh only a subset of any Reactor’s channels, and may then use a correspondingly increased refresh rate.
Note that there is no corresponding rate info for the PSI Master, since all messages, except for Discoveries, are explicitly

requested by the PSI Master, which therefore controls the rate.
It’s value is "bit 24".

Query maximum number of GIDs: PSI_NO_FM_MGIDREQ

This flag is used to request transmission of the maximum number of GIDs that can be assigned to a Reactor.
It’s value is "bit 23".

Query GIDs: PSI_NO_FM_GIDREQ

This flag is used to request transmission of all GIDs assigned to it to the PSI Master.
It’s value is "bit 22".

Query vendor specific information: PSI_NO_FM_VSIREQ

This flag is used to request transmission of vendor specific information like vendor name, device name, model number, etc. to the PSI Master. The Reactor sends the vendor string (using the generic text message Word PSI_ST_TM_WTP), the vendor numbers (using the generic 32 bit Word PSI_ST_TM_WG32), and the vendor MIME data (using the generic MIME type word PSI_ST_TM_WMP), if any. If nothing has been set, an empty sentence of the respective type is sent.
It’s value is "bit 21".

Query user specific information: PSI_NO_FM_USIREQ

This flag is used to request transmission of all user specific information. Like PSI_NO_FM_VSIREQ, the Reactor sends the user string (using the generic text message Word PSI_ST_TM_WTP), the user numbers (using the generic 32 bit Word PSI_ST_TM_WG32), and the user MIME data (using the generic MIME type word PSI_ST_TM_WMP), if any. If nothing has been set, an empty sentence of the respective type is sent.
It’s value is "bit 20".

Query the Reactor’s type: PSI_NO_FM_RTREQ

This flag is used to request transmission of the Reactor’s type to the PSI Master.
It’s value is "bit 19".

Query the Reactor’s Identification Number: PSI_NO_FM_INREQ

This flag makes the Reactor send a message. Since the IN is part of the message header (see Section 3.5.1), any message that needs to be sent to the PSI Master anyway will do. If no message needs to be sent at that time, an empty message of type PSI_ST_TM_WN is sent.
It’s value is "bit 18".

Query channel counts: PSI_NO_FM_CCREQ

This flag is used to request transmission of the number of channels of each type (Output, InOut, Input) to the PSI Master. Channels using Safety Sequence count as Output or InOut, according to their properties.
It’s value is "bit 17".

Request to use the Current Values: PSI_NO_FM_GODATA

If this flag is set, and the Reactor is currently using Safe Values, it will start using the Current Values as soon as possible, usually after reception of values for every channel. In any case, a Reactor MUST NOT start using Current Values if a timeout occurred, or if it has not actually received values, or the values received are not valid (anymore). If it is sent in a message containing data Words, then the values received are applied immediately. If the same message has the flag PSI_NO_FM_GOSAFE set, then that MUST be given precedence. PSI_SO_FM_GOSAFE, if set in this message or previously, overrides this flag on a per-channel basis, but does not clear it. Therefore, PSI_NO_FM_GODATA applies to these channels as soon as PSI_SO_FM_GODATA is received, unless it has been cleared meanwhile by PSI_NO_FM_GOSAFE or a timeout occurred. As an exception, channels using Safety Sequence MUST discard received data unless they are set to use Current Values. This flag may be sent at any time, but is restricted to Output and InOut Reactors.
It’s value is "bit 16".

Note: because there is no guarantee that any message sent actually reaches the receiver, this flag SHOULD be repeated as long as it is supposed to be in effect. This can easily be done by setting this flag in all Node Sections. Alternatively, the Reactor’s status may be queried, or replies that can doubtlessly be attributed to requests having been sent in this same sentence may also be taken as acknowledgement. Furthermore, the flag PSI_NO_FM_SENDACK may also be used.

Request to use the Safe Values: PSI_NO_FM_GOSAFE

If this bit is set, all Output and InOut channels MUST be switched to their respective Safe Values, regardless of the remaining content of the sentence or other Sentence Options. This MUST only be reverted upon reception of PSI_NO_FM_GODATA. This flag MUST be applied regardless of the state of the channels, even if they have already been individually set to use Safe Values. This means that when a channel gets set to use Current Values by use of PSI_SO_FM_GODATA, it will still continue using Safe Values until PSI_NO_FM_GODATA is sent to it's Reactor. Channels using Safety Sequence MUST clear any progress made through their sequence, and cease operation or otherwise ensure safety if at all possible.
This flag SHOULD be used whenever a PSI Master cannot generate new values for any channel of that Reactor, for example if input channels that are used to generate the values cease providing data. It MUST be used if the PSI Master detects an internal or external problem that could result in incorrect values being generated.
It may also be used prior to shutting down the PSI Master, or before a Reactor is removed from the PSI Master's inventory.
It is valid for all Reactor types and may be sent at any time and combined with all other flags; if PSI_NO_FM_GODATA is set then that is ignored, PSI_NO_FM_GOSAFE always taking precedence. Input Reactors ignore this flag.
It's value is "bit 15".

Note: because there is no guarantee that any message sent actually reaches the receiver, this flag must be repeated as long as it is supposed to be in effect. This can easily be done by setting this flag in all Node Sections. Alternatively, the Reactor's status may be queried, or the flag PSI_NO_FM_SENDACK may be used, so a correspondingly set PSI_NO_TM_ACK or PSI_NO_TM_NACK suffices.

Node Section applies to all Group members: PSI_NO_FM_CLUSTER

This flag is used to indicate that the current Node Section is to be interpreted by all receiving Reactors that are members of the Group specified, and MUST only be used in messages of type multicast. A Node Section that has this flag set has no TIN field. This flag exists to increase efficiency in case there are several Reactors that are supposed to receive identical data. Normally, this will be the case for functionally similar Reactors, but because it may be combined with any flag except PSI_NO_FM_ACK, PSI_NO_FM_NACK, PSI_NO_FM_REACTOR_ACCEPTED and PSI_NO_FM_NODE_FINISHED, it is not limited to sending data (see Section 2.2). While Reactors MUST implement it, implementation

and use of this flag MAY be omitted for PSI Masters.
It’s value is "bit 14".

Reply to a Discovery message: PSI_NO_FM_REACTOR_ACCEPTED

If this flag is set, the PSI Master has added the Reactor to it’s inventory. The PSI Master sends a message with this flag set for each Discovery Message it receives, until the Reactor ceases sending further Discoveries. This ensures that even if an arbitrary number of Discovery messages and / or replies get lost, every Reactor in the Nexus is known to the PSI Master and has received a reply in the end.
It’s value is "bit 13".

This means that

* prior to receipt of a reply by a Reactor, the PSI Master has received at least one Discovery message from that Reactor and therefore knows of it and that

* upon arrival of a Discovery message at the PSI Master the sending Reactor may not yet have received a reply, or the Discovery may already have been on it’s way before the Reactor received the reply. Since this cannot be known, another reply must be sent. See Section 3.8 for details.

Request to send a confirmation: PSI_NO_FM_SENDACK

The receiving Reactor MUST acknowledge, after having executed them, all Node Sections that have this flag set. This is done by sending each of them back (keeping the FM types) inside a Node Section that has either of the flags PSI_NO_TM_NACK and PSI_NO_TM_ACK set. The Reactor decides whether to use NACK or ACK, and normally the decision should be based on whichever results in the shortest message. Therefore, it is expected that most of the time NACK will be used, because if nothing failed only an empty message needs to be sent. The PSI Master MUST ensure that the resulting acknowledgement will fit into it’s maximum message length.
Using this flag may be especially worthwhile in case of channels using Safety Sequence, so the advancement can be monitored, and is the basis of the ACKed mode (see Section 2.6).
This flag must not be combined with the flags PSI_NO_FM_ACK and PSI_NO_FM_NACK.
It’s value is "bit 12".

Confirmation of executing a request: PSI_NO_FM_ACK

This flag indicates that the PSI Master has received and executed one or more requests by a Reactor. It is always used in conjunction with the Request ID field, like PSI_NO_FM_NACK. The Node Section with this flag set contains all requests and data that were executed or used from the Node Section that contained the PSI_NO_TM_SENDACK flag. The first sentence MUST be of type PSI_Word_Node_Specification and contain a single Word that holds all flags from the original Node Options field that were not executed or used (keeping the TM types). Everything that was in the original Node Section but is missing from the one with this flag set was not executed. This kind of acknowledge is sent when PSI_NO_TM_SENDACK is set, and each Node Section that has that flag set is acknowledged separately, they are never split nor conglomerated. The Reactor therefore MUST ensure that the resulting acknowledgement fits into it’s maximum message length, and split the requests if needed. Multiple Node Sections of the same message may have PSI_NO_TM_SENDACK set, and the individual acknowledgements may be sent in different messages, though it is recommended to use a single message if possible.
This flag must not be combined with the flags PSI_NO_FM_NACK and PSI_NO_FM_SENDACK.
It’s value is "bit 11".

Notification of not executing a request: PSI_NO_FM_NACK

This flag indicates that the PSI Master has received but not executed one or more requests by a Reactor. It is always used in conjunction with the Request ID field, like PSI_NO_FM_ACK. The Node Section with this flag set contains all requests and data that were not executed or used from the Node Section that contained the PSI_NO_TM_SENDACK flag. The first sentence MUST be of type PSI_Word_Node_Specification and contain a single Word that holds all flags from the original Node Options field that were not executed or used (keeping the TM types). Everything that was in the original Node Section but is missing from the one with this flag set was executed. This kind of negative acknowledge is sent when PSI_NO_TM_SENDACK is set, and each Node Section that has that flag set is negatively acknowledged separately, they are never split nor conglomerated. The Reactor therefore MUST ensure that the resulting acknowledgement fits into it’s maximum message length, and split the requests if needed. Multiple Node Sections of the same message may have PSI_NO_TM_SENDACK set, and the individual acknowledgements may be sent in different messages, though it is recommended to use a single message if possible.
This flag must not be combined with the flags PSI_NO_FM_ACK and

PSI_NO_FM_SENDACK.
It’s value is "bit 10".

End of relevant message content: PSI_NO_FM_NODE_FINISHED

If this flag is set in a Node Section it indicates that this is the last Node Section containing data for the specific Reactor. The Reactor therefore does not need to interpret the remainder of the message, if any. However, for error detection, the entire message still needs to be checked, meaning that it must have been received in full and the length data must match each other. If this is not the case, then the message must be discarded and no information or requests contained in it must be used or executed because it is not clear which part the error actually affects.
It’s value is "bit 9".

Node Options for messages to the PSI Master:

The following Node Options are valid going to the PSI Master. With certain exceptions, they may be combined arbitrarily.

Reactor status OK: PSI_NO_TM_SOK

If this flag is set, there are no issues at all. It therefore must only be set if none of the other Reactor status bits (PSI_NO_TM_Sx) is set.
It’s value is "bit 31".

Non-critical information: PSI_NO_TM_SNCI

If this flag is set, there is non-critical information, so a status inquiry is recommended. The correct functioning of the Reactor is not affected. This is the case, for example, when the Reactor has been switched to Safe Values. If this flag is set, PSI_NO_TM_SOK must not be set.
It’s value is "bit 30".

Non-critical error: PSI_NO_TM_SNCE

If this flag is set, there is a non-critical error, so a status inquiry is strongly recommended. This is the case, for example, if the Reactor has started using Safe Values due to a timeout. If this flag is set, PSI_NO_TM_SOK must not be set.
It’s value is "bit 29".

Critical error: PSI_NO_TM_SCE

If this flag is set, there is a critical error that may affect the Reactor's operation and possibly damage it. Therefore, a status inquiry is required to query the exact problem(s). This is the case, for example, if channels stop functioning or another hardware or software error has been detected. If this flag is set, PSI_NO_TM_SOK must not be set.
It's value is "bit 28".

Status unchanged: PSI_NO_TM_SUNCH

The Reactor's status is not OK, but has not changed since the last status inquiry; another inquiry is not required, but allowed at any time. If this flag is set, then PSI_NO_TM_SOK must not be set.
It's value is "bit 27".

Request to send a confirmation: PSI_NO_TM_SENDACK

The PSI Master MUST confirm all Node Sections marked with this flag after having executed them. This is done by sending each of them back (keeping the TM types) inside a Node Section that has either of the flags PSI_NO_FM_NACK and PSI_NO_FM_ACK set. The PSI Master decides whether to use NACK or ACK, and normally the decision should be based on whichever results in the shortest message. Therefore, it is expected that most of the time NACK will be used, because if nothing failed only an empty message needs to be sent. The Reactor MUST ensure that the resulting acknowledgement fits into it's maximum message length.
This flag must not be combined with the flags PSI_NO_TM_ACK and PSI_NO_TM_NACK.
It's value is "bit 26".

Confirmation of executing a request: PSI_NO_TM_ACK

This flag indicates that the Reactor has received and executed one or more requests by the PSI Master. It is always used in conjunction with the Request ID field, like PSI_NO_TM_NACK. The Node Section with this flag set contains all requests and data that were executed or used from the Node Section that contained the PSI_NO_FM_SENDACK flag. The first sentence MUST be of type PSI_Word_Node_Specification and contain a single Word that holds all flags from the original Node Options field that were executed or used (keeping the FM types). Everything that was in the

original Node Section but is missing from the one with this flag set was not executed. This kind of acknowledge is sent when PSI_NO_FM_SENDACK is set, and each Node Section that has that flag set is acknowledged separately, they are never split nor conglomerated. The PSI Master therefore MUST ensure that the resulting acknowledgement fits into it's maximum message length, and split the requests if needed. Multiple Node Sections in the same message may have PSI_NO_FM_SENDACK set, and the individual acknowledgements may be sent in different messages, though it is recommended to use a single message if possible.
This flag must not be combined with the flags PSI_NO_TM_NACK and PSI_NO_TM_SENDACK.
It's value is "bit 25".

Notification of not executing a request: PSI_NO_TM_NACK

This flag indicates that the Reactor has received but not executed one or more requests by the PSI Master. It is always used in conjunction with the Request ID field, like PSI_NO_TM_ACK. The Node Section with this flag set contains all requests and data that were not executed or used from the Node Section that contained the PSI_NO_FM_SENDACK flag. The first sentence MUST be of type PSI_Word_Node_Specification and contain a single Word that holds all flags from the original Node Options field that were not executed or used (keeping the FM types). Everything that was in the original Node Section but is missing from the one with this flag set was executed. This kind of negative acknowledge is sent when PSI_NO_FM_SENDACK is set, and each Node Section that has that flag set is negatively acknowledged separately, they are never split nor conglomerated. The PSI Master therefore MUST ensure that the resulting acknowledgement fits into it's maximum message length, and split the requests if needed. Multiple Node Sections in the same message may have PSI_NO_FM_SENDACK set, and the individual acknowledgements may be sent in different messages, though it is recommended to use a single message if possible.
This flag must not be combined with the flags PSI_NO_TM_ACK and PSI_NO_TM_SENDACK.
It's value is "bit 24".

End of relevant message content: PSI_NO_TM_MASTER_FINISHED

If this flag is set in a Node Section it indicates that this is the last Node Section containing data for the specific PSI Master. The PSI Master therefore does not need to interpret the remainder of the message, if any. However, for error detection, the entire message still needs to be checked, meaning that it must have been

received in full and the length data must match each other. If this is not the case, then the message must be discarded and no information or requests contained in it must be used or executed because it is not clear which part the error actually affects. It's value is "bit 23".

3.7.6. Sentence Options

This section describes the constants defined for Sentence Options. The Sentence Options are encoded as bit field to conserve bandwidth. The definitions of channel number sizes apply to all Words within any given sentence. Every node MUST be able cope with all sizes, but the smallest size possible to SHOULD always be used. Especially, if there is a substantial number of channels below any specific size definition in the same sentence as one or more above that size definition, they SHOULD be split into separate sentences so the smaller type can be used for the lower channels.

Sentence Options for messages from the PSI Master:

These Sentence Options are valid coming from the PSI Master. They may be combined in any number, with certain exceptions. This way it is possible to request all possible information about channels using a single message.

Defines channel numbers to be 8 bit: PSI_SO_FM_CN8

This bit defines the size of all channel numbers in the sentence to be 8 bit in size. It may only be used in conjunction with Words actually containing channel numbers, and must not be combined with any other channel number definition.
It's value is "bit 15".

Defines channel numbers to be 16 bit: PSI_SO_FM_CN16

This bit defines the size of all channel numbers in the sentence to be 16 bit in size. It may only be used in conjunction with Words actually containing channel numbers, and must not be combined with any other channel number definition.
It's value is "bit 14".

Defines channel numbers to be 32 bit: PSI_SO_FM_CN32

This bit defines the size of all channel numbers in the sentence to be 32 bit in size. It may only be used in conjunction with Words actually containing channel numbers, and must not be

combined with any other channel number definition.
It’s value is "bit 13".

Query channel status: PSI_SO_FM_CSREQ

When this bit is set, the Reactor will send the status of the channels given in the sentence. Normally, it is used in sentences of type Channel Number, but it may be used with any type that contains a channel number. It is valid for all channel types and may be sent at any time.
It’s value is "bit 12".

Query the data types used: PSI_SO_FM_DTREQ

When this bit is set, the Reactor will send the data types (that define the Word type to be used) of the channels given in the sentence. The data type returned is the one that the channel sends or receives, so an Input channel will return the TM type while an Output channel will return the FM type here. An InOut channel may return either.
It may be used only in Words of type Channel Number, because the type of Data Word to be used is not known yet. Likewise, the use of the flag PSI_SO_FM_GODATA is not allowed, because for lack of knowledge of the data types, no data can have been sent. Use of the flag PSI_SO_FM_GOSAFE is allowed. This Option is valid for all channel types and may be sent at any time. Because a channel using Safety Sequence always uses an additional data Word, the type of that needs to be returned. A channel using Safety Sequence MUST discard any progress through the sequence if this Option is received, and ensure safety if at all possible.
While it may seem tempting, it it DISCOURAGED to have one channel use multiple Word types (for example an integer and the time type). While this option is slightly better than using the generic Word types, and is OPTIONAL for any PSI Master, it is RECOMMENDED for Reactors to always use distinct channels even if they logically represent a single function (data and delay, for example), because this can be used by generic PSI Masters. A PSI Master that does not support it should not access channels returning multiple Word types. Given that channel numbers are per-Reactor, multiplying the number of channels of a Reactor to avoid multiple Word types has no negative effect.
It’s value is "bit 11".

Note: Replies to PSI_SO_FM_VREQ allow deducing the Word type used, so using this flag is not strictly necessary.

Query Safety Sequence: PSI_SO_FM_SSREQ

When this bit is set, the Reactor will send the Safety Sequences for the channels given in the sentence. Because the sequence is not supposed to be known yet, it may only be used with Words of type Channel Number. It's use is restricted to channels using Safety Sequence and may be sent at any time. A channel using Safety Sequence MUST discard any progress through the sequence if this Option is received, and also cease operation and ensure safety if at all possible.
It's value is "bit 10".

Query channel type: PSI_SO_FM_CTREQ

If this bit is set, the Reactor will send the types of the channels given in the sentence. Because the channel details cannot be known yet, it may only be used with Words of type Channel Number. The use of PSI_SO_FM_GODATA is not allowed in this sentence, since no data values can have been sent yet. Use of PSI_SO_FM_GOSAFE is allowed. It may be used with all channel types and may be sent at any time.
It's value is "bit 9".

Query the channel's data boundaries: PSI_SO_FM_DBREQ

If this bit is set, the Reactor sends the minimum and maximum values that are meaningful to the channels listed in the sentence. This bit may be used with any Word type that contains a channel number, and may be sent to channels of all types and at any time. Even though it is expected that for most channels the boundaries will be given by the Word type, this query must be used because that is not necessarily the case. Knowing the proper range is important for correctly calculating values to send, especially since a receiver may use any method for fitting the data received into it's storage (discarding the superfluous bits is suggested).
It's value is "bit 8".

Value request: PSI_SO_FM_VREQ

If this bit is set, the Reactor will send the Current Values of the channels given in the sentence (for output channels, the value assigned last is sent). Normally, it is used in sentences of type Channel Number, but it may be used with any type that contains a channel number. It is valid for all channel types and may be sent at any time.

It’s value is "bit 7".

Maximum message length information: PSI_SO_FM_MMLINFO

This flag is valid only in conjunction with a Word of type PSI_Word_Node_Specification, that contains the maximum message length of the PSI Master. A Reactor MUST NOT send messages larger than that to the PSI Master, but may send any size below. The PSI Master MAY change this at any time during normal operation, usually as part of adapting to packet loss. It then sends a message with this flag, and usually also the flag PSI_NO_FM_SENDACK set. This way, and because the PSI Master controls the length of it’s own messages, it is in full control of the maximum message sizes that are sent, and therefore the Reactors do not need to be reconfigured to adapt to new network conditions.
The PSI Master’s maximum message length SHOULD be around 1400 octets, at least initially, to allow configuration information to be sent as complete as possible, but MAY become significantly larger and also smaller in response to network packet loss.
This means that Words without fixed length, like vendor- and user-specific information, should be well below 1400 octets in length, because otherwise they may be truncated, or sending may abort with an error. Don’t be unnecessarily chatty! ;)
It’s value is "bit 6".

Assignment of Group IDs: PSI_SO_FM_GIDSET

This bit may only be used in conjunction with Words of type PSI_Word_Node_Specification, that contain the GIDs to be used. Multiple Words may be used to assign as many GIDs. Any sentence having this flag set clears and replaces all GIDs assigned previously, if any. Use of this flag in an empty sentence removes all assigned GIDs. Execution must be acknowledged (see PSI_NO_TM_SENDACK in Section 3.7.5). Use of this flag is optional.
It’s value is "bit 5".

Additional assignment of Group IDs: PSI_SO_FM_GIDADD

This bit may only be used in conjunction with Words of type PSI_Word_Node_Specification, that contain the GIDs to be added. Multiple Words may be used to add as many GIDs. Any sentence having this flag set adds the GIDs listed to any GIDs that were assigned previously. Execution must be acknowledged (see

PSI_NO_TM_SENDACK in Section 3.7.5). Because of duplicated packets and lost ACKs, duplicates may occur, so the receiver must be prepared to handle this. These cases are no errors: the request must be acknowledged as if no duplicate had occurred. Use of this flag is optional.
It's value is "bit 4".

Removal of Group IDs: PSI_SO_FM_GIDREM

This bit may only be used in conjunction with Words of type PSI_Word_Node_Specification, that contain the GIDs to be removed. Multiple Words may be used to remove as many GIDs. Any duplicates, if they were not weeded out already, must also be removed from the list. Use of this flag in an empty sentence removes all assigned GIDs. Execution must be acknowledged (see PSI_NO_TM_SENDACK in Section 3.7.5). Because of duplicated packets and lost ACKs, requests to remove GIDs that are not assigned to the Reactor may occur, so it must be prepared to handle this. These cases are not errors: the request must be acknowledged as if no duplicate had occurred. Use of this flag is optional.
It's value is "bit 3".

Request to use Current Values: PSI_SO_FM_GODATA

If this flag is set, and the channels listed are currently using Safe Values, they will start using the Current Values as soon as possible, usually after reception of the first value. In any case, a channel MUST NOT start using Current Values if a timeout occurred, or if it has not actually received values, or the values received are not valid (anymore). This flag may be used with any Word type containing a channel number, and at any time, but is restricted to Output and InOut channels. If it is set in a sentence containing data Words, then the values received are applied immediately. If the same sentence has the flag PSI_SO_FM_GOSAFE set, then that MUST be given precedence. PSI_NO_FM_GOSAFE, if set in this message or previously, overrides this flag, but does not clear it. Therefore, PSI_SO_FM_GODATA is applied as soon as PSI_NO_FM_GODATA is received, unless it has been cleared meanwhile by PSI_SO_FM_GOSAFE or a timeout occurred. As an exception, channels using Safety Sequence MUST discard received data unless they are set to use Current Values.
It's value is "bit 2".

Note: because there is no guarantee that any message sent actually reaches the receiver, this flag SHOULD be repeated as long as it

is supposed to be in effect. This can easily be done by setting this flag in all sentences. Alternatively, the channels status may be queried, or replies that can doubtlessly be attributed to requests having been sent in this same sentence may also be taken as acknowledgement. Furthermore, the flag PSI_NO_FM_SENDACK may also be used.

Request to use Safe Values: PSI_SO_FM_GOSAFE

If this bit is set, all channels listed in this sentence MUST be switched to their respective Safe Values, regardless of the remaining content of the sentence or other Sentence Options. This MUST only be reverted upon reception of PSI_SO_FM_GODATA. This flag MUST be applied regardless of the state of the Reactor, even if the Reactor as a whole has already been set to use Safe Values. This means that when the Reactor gets set to use Current Values by use of PSI_NO_FM_GODATA, channels having received this flag continue using Safe Values until PSI_SO_FM_GODATA is sent to them. Channels using Safety Sequence MUST clear any progress made through their Safety Sequence, and cease operation or otherwise ensure safety if at all possible. This flag MUST be used whenever a PSI Master cannot generate new values for a channel, for example if an Input channel that is used to generate the value ceases providing data. It is valid for all channel types (Input channels ignore it) and may be sent at any time and combined with all other flags; if PSI_SO_FM_GODATA is set then that is ignored, PSI_SO_FM_GOSAFE always taking precedence.
It’s value is "bit 1".

Note: because there is no guarantee that any message sent actually reaches the receiver, this flag must be repeated as long as it is supposed to be in effect. This can easily be done by setting this flag in all sentences. Alternatively, the channel’s status may be queried, or the flag PSI_NO_FM_SENDACK may be used, so a correspondingly set PSI_NO_TM_ACK or PSI_NO_TM_NACK suffices.

Assignment of new values: PSI_SO_FM_VSET

This flag must only be used in sentences containing data Words and makes the Reactor assign the new values to the corresponding channels. Except for channels using Safety Sequence, this applies regardless of the current state of the Reactor: the values are applied even if the Reactor or channels are set to use Safe Values. As soon as these conditions are removed, the values are used as long as no timeout occurs. Channels using Safety Sequence MUST discard any values sent to them unless they are set to use

Current Values and have no other prohibitive states. This flag is valid for Output and InOut channels only.
It’s value is "bit 0".

Sentence Options for messages to the PSI Master:

These Sentence Options are valid in messages sent to the PSI Master. Contrary to those coming from the PSI Master, and except for being combined with a channel number definition, they are mutually exclusive because they define the meaning of generic Words.

Defines channel numbers to be 8 bit: PSI_SO_TM_CN8

This bit defines the size of all channel numbers in the sentence to be 8 bit in size. It may only be used in conjunction with Words actually containing channel numbers, and must not be combined with any other channel number definition.
It’s value is "bit 15".

Defines channel numbers to be 16 bit: PSI_SO_TM_CN16

This bit defines the size of all channel numbers in the sentence to be 16 bit in size. It may only be used in conjunction with Words actually containing channel numbers, and must not be combined with any other channel number definition.
It’s value is "bit 14".

Defines channel numbers to be 32 bit: PSI_SO_TM_CN32

This bit defines the size of all channel numbers in the sentence to be 32 bit in size. It may only be used in conjunction with Words actually containing channel numbers, and must not be combined with any other channel number definition.
It’s value is "bit 13".

Words contain Data Type information: PSI_SO_TM_DTINFO

This flag means that the Words of type PSI_Word_Channel_Specification in this sentence contain the data types used by the channels listed. These need to be known before values can be sent or received.
It’s value is "bit 12".

Words contain Channel Type information: PSI_SO_TM_CTINFO

This flag means that the Words of type PSI_Word_Channel_Specification in this sentence contain the types of the channels listed. These need to be known before values can be sent or received.
It's value is "bit 11".

Words contain data boundary information: PSI_SO_TM_DBINFO

This flag means that the Words in this sentence contain the boundaries of the channels listed. These should be known before values are sent or received.
In order to prevent proliferation of Word types, the same types as used for data values are used. Therefore for each channel, two data Words of the same type are sent consecutively within the same sentence, both containing the channel number. The first Word contains the minimum value and the second one contains the maximum value in the value syllable. The overhead created by sending the channel number a second time is tolerable because this kind of sentence is used only upon initialization of the Nexus and thus does not impact normal operation.
Channels for which a range cannot be meaningfully defined are those using the string and MIME data types and possibly the opaque type, depending on the actual content of the opaque data. These channels send the minimum and maximum data sizes inside two instances of the generic Channel Specification Word type.
Channels using the MIME data type additionally send a Word of type String that contains all supported MIME types, one per line.
It's value is "bit 10".

Words contain Current Values: PSI_SO_TM_VINFO

This flag means that the Words in this sentence contain the Current Values of the respective channels, of the appropriate data type.
It's value is "bit 9".

Maximum message length information: PSI_SO_TM_MMLINFO

This flag is valid only in conjunction with a Word of type PSI_Word_Node_Specification, that contains the maximum message length of the Reactor. A PSI Master MUST NOT send messages larger than that to the Reactor, but may send any size below. Even though not normally necessary, the Reactor MAY change this during

normal operation, usually after user configuration changes. In that case, it MUST set the status PSI_RS_REREAD_ALL so the PSI Master will request the information anew.
The maximum message length SHOULD be at least around 1400 octets, but MAY be significantly larger.
It’s value is "bit 8".

Maximum refresh rate information: PSI_SO_TM_MRRINFO

This flag is valid only in conjunction with a sentence containing one or two Words of type PSI_Word_Node_Specification, that contain the maximum refresh rates of the Reactor. The first Word contains the maximum refresh rate for normal channels, while the second Word contains the maximum refresh rate for all channels using Safety Sequence, and is present only if the Reactor actually contains any such channels.
The Reactor MAY change these during normal operation, usually after user configuration changes. In that case, it MUST set the status PSI_RS_REREAD_ALL so the PSI Master will request the information anew.
The maximum refresh rate for normal channels MUST be at least 100 Hz, but MAY be significantly larger. That of channels using Safety Sequence is unrestricted and depends solely on the needs of the respective channels.
It’s value is "bit 7".

Word contains GID maximum: PSI_SO_TM_MGIDINFO

This flag means that the Word of type PSI_Word_Node_Specification in this sentence contains the maximum number of GIDs that can be assigned to the Reactor. At minimum, a Reactor should be able to be assigned 10 GIDs.
It’s value is "bit 6".

Word contains Reactor type: PSI_SO_TM_RTINFO

This flag means that the Word of type PSI_Word_Node_Specification in this sentence contains the type of the Reactor. This needs to be known before values can be sent or received.
It’s value is "bit 5".

Words contain Group identifiers: PSI_SO_TM_GIDINFO

This flag means that the Words of type PSI_Word_Node_Specification in this sentence contain the GIDs that currently are assigned to the Reactor. These need to be known before multicast messages can be sent.
It’s value is "bit 4".

Words contain vendor specific information: PSI_SO_TM_VSINFO

This flag means that the Words in this sentence contain vendor specific information; the sentence type is either PSI_ST_TM_WTP, PSI_ST_TM_WG32 or PSI_ST_TM_WMP.
It’s value is "bit 3".

Words contain user specific information: PSI_SO_TM_USINFO

This flag means that the Words in this sentence contain user specific information; the sentence type is either PSI_ST_TM_WTP, PSI_ST_TM_WG32 or PSI_ST_TM_WMP.
It’s value is "bit 2".

3.7.7. Sentence types

The sentence type defines the type of the Words that the sentence consists of.

Sentence types for messages from a PSI Master:

Sentence of boolean Data Words: PSI_ST_FM_WDB

This type identifies a sentence consisting of Words of type PSI_Word_DatumB.
It’s value is (PSI_ST_BASE_FM+0).

Sentence of signed 8 bit Data Words: PSI_ST_FM_WDS8

This type identifies a sentence consisting of Words of type PSI_Word_DatumS8.
It’s value is (PSI_ST_BASE_FM+1).

Sentence of unsigned 8 bit Data Words: PSI_ST_FM_WDU8

This type identifies a sentence consisting of Words of type PSI_Word_DatumU8.
It’s value is (PSI_ST_BASE_FM+2).

Sentence of signed 16 bit Data Words: PSI_ST_FM_WDS16

This type identifies a sentence consisting of Words of type PSI_Word_DatumS16.
It’s value is (PSI_ST_BASE_FM+3).

Sentence of unsigned 16 bit Data Words: PSI_ST_FM_WDU16

This type identifies a sentence consisting of Words of type PSI_Word_DatumU16.
It’s value is (PSI_ST_BASE_FM+4).

Sentence of signed 32 bit Data Words: PSI_ST_FM_WDS32

This type identifies a sentence consisting of Words of type PSI_Word_DatumS32.
It’s value is (PSI_ST_BASE_FM+5).

Sentence of unsigned 32 bit Data Words: PSI_ST_FM_WDU32

This type identifies a sentence consisting of Words of type PSI_Word_DatumU32.
It’s value is (PSI_ST_BASE_FM+6).

Sentence of signed 64 bit Data Words: PSI_ST_FM_WDS64

This type identifies a sentence consisting of Words of type PSI_Word_DatumS64.
It’s value is (PSI_ST_BASE_FM+7).

Sentence of unsigned 64 bit Data Words: PSI_ST_FM_WDU64

This type identifies a sentence consisting of Words of type PSI_Word_DatumU64.
It’s value is (PSI_ST_BASE_FM+8).

Sentence of floating-point Data Words: PSI_ST_FM_WDF32

This type identifies a sentence consisting of Words of type PSI_Word_DatumF32.
It’s value is (PSI_ST_BASE_FM+9).

Sentence of double floating-point Data Words: PSI_ST_FM_WDF64

This type identifies a sentence consisting of Words of type PSI_Word_DatumF64.
It’s value is (PSI_ST_BASE_FM+10).

Sentence of quadruple floating-point Data Words: PSI_ST_FM_WDF128

This type identifies a sentence consisting of Words of type PSI_Word_DatumF128.
It’s value is (PSI_ST_BASE_FM+11).

Sentence of fixed-point Data Words with 16 bit divisor: PSI_ST_FM_WDFIPOD16

This type identifies a sentence consisting of Words of type PSI_Word_Datum_Fixedpoint_D16.
It’s value is (PSI_ST_BASE_FM+12).

Sentence of fixed-point Data Words with 64 bit divisor: PSI_ST_FM_WDFIPOD64

This type identifies a sentence consisting of Words of type PSI_Word_Datum_Fixedpoint_D64.
It’s value is (PSI_ST_BASE_FM+13).

Sentence of one opaque Data Word with variable length: PSI_ST_FM_WDO

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_Data_Opaque.
It’s value is (PSI_ST_BASE_FM+14).

Sentence of one text Data Word with variable length: PSI_ST_FM_WDT

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_Data_Text.

It’s value is (PSI_ST_BASE_FM+15).

Sentence of one MIME Data Word with variable length: PSI_ST_FM_WDM

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_Data_MIME.
It’s value is (PSI_ST_BASE_FM+16).

Sentence of time Data Words: PSI_ST_FM_W_T

This type identifies a sentence consisting of Words of type PSI_Word_Data_Time.
It’s value is (PSI_ST_BASE_FM+17).

Sentence of one Safety Sequence Word of variable length: PSI_ST_FM_WSS

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_Safety_Sequence. The sentence length in this direction MUST be 1.
It’s value is (PSI_ST_BASE_FM+18).

Sentence of Channel Numbers: PSI_ST_FM_WCN

This type identifies a sentence consisting of Words of type PSI_Word_Channel_Number.
It’s value is (PSI_ST_BASE_FM+19).

Sentence of generic Channel Specification Words: PSI_ST_FM_W_C_S

This type identifies a sentence consisting of Words of type PSI_Word_Channel_Specification.
It’s value is (PSI_ST_BASE_FM+20).

Sentence of one plain opaque Word with variable length: PSI_ST_FM_WOP

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_Opaque_Plain.
It’s value is (PSI_ST_BASE_FM+21).

Sentence of one plain text Word with variable length: PSI_ST_FM_WTP

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_Text_Plain.
It's value is (PSI_ST_BASE_FM+22).

Sentence of one plain MIME Word with variable length: PSI_ST_FM_WMP

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_MIME_Plain.
It's value is (PSI_ST_BASE_FM+23).

Sentence of plain time Words: PSI_ST_FM_W_T_P

This type identifies a sentence consisting of Words of type PSI_Word_Time_Plain.
It's value is (PSI_ST_BASE_FM+24).

Sentence of generic 32 bit Words: PSI_ST_FM_WG32

This type identifies a sentence consisting of Words of type PSI_Word_Generic_32.
It's value is (PSI_ST_BASE_FM+25).

Sentence of generic Node Specification Words: PSI_ST_FM_W_N_S

This type identifies a sentence consisting of Words of type PSI_Word_Node_Specification.
It's value is (PSI_ST_BASE_FM+26).

Sentence without content: PSI_ST_FM_WN

This type identifies a sentence containing no Words. It's main purpose is for debugging.
It's value is (PSI_ST_BASE_FM+27).

Sentence types for messages to a PSI Master:

Sentence of boolean Data Words: PSI_ST_TM_WDB

This type identifies a sentence consisting of Words of type PSI_Word_DatumB.
It's value is (PSI_ST_BASE_TM+0).

Sentence of signed 8 bit Data Words: PSI_ST_TM_WDS8

This type identifies a sentence consisting of Words of type PSI_Word_DatumS8.
It's value is (PSI_ST_BASE_TM+1).

Sentence of unsigned 8 bit Data Words: PSI_ST_TM_WDU8

This type identifies a sentence consisting of Words of type PSI_Word_DatumU8.
It's value is (PSI_ST_BASE_TM+2).

Sentence of signed 16 bit Data Words: PSI_ST_TM_WDS16

This type identifies a sentence consisting of Words of type PSI_Word_DatumS16.
It's value is (PSI_ST_BASE_TM+3).

Sentence of unsigned 16 bit Data Words: PSI_ST_TM_WDU16

This type identifies a sentence consisting of Words of type PSI_Word_DatumU16.
It's value is (PSI_ST_BASE_TM+4).

Sentence of signed 32 bit Data Words: PSI_ST_TM_WDS32

This type identifies a sentence consisting of Words of type PSI_Word_DatumS32.
It's value is (PSI_ST_BASE_TM+5).

Sentence of unsigned 32 bit Data Words: PSI_ST_TM_WDU32

This type identifies a sentence consisting of Words of type PSI_Word_DatumU32.
It's value is (PSI_ST_BASE_TM+6).

Sentence of signed 64 bit Data Words: PSI_ST_TM_WDS64

This type identifies a sentence consisting of Words of type PSI_Word_DatumS64.
It’s value is (PSI_ST_BASE_TM+7).

Sentence of unsigned 64 bit Data Words: PSI_ST_TM_WDU64

This type identifies a sentence consisting of Words of type PSI_Word_DatumU64.
It’s value is (PSI_ST_BASE_TM+8).

Sentence of floating-point Data Words: PSI_ST_TM_WDF32

This type identifies a sentence consisting of Words of type PSI_Word_DatumF32.
It’s value is (PSI_ST_BASE_TM+9).

Sentence of double floating-point Data Words: PSI_ST_TM_WDF64

This type identifies a sentence consisting of Words of type PSI_Word_DatumF64.
It’s value is (PSI_ST_BASE_TM+10).

Sentence of quadruple floating-point Data Words: PSI_ST_TM_WDF128

This type identifies a sentence consisting of Words of type PSI_Word_DatumF128.
It’s value is (PSI_ST_BASE_TM+11).

Sentence of fixed-point Data Words with 16 bit divisor: PSI_ST_TM_WDFIPOD16

This type identifies a sentence consisting of Words of type PSI_Word_Datum_Fixedpoint_D16.
It’s value is (PSI_ST_BASE_TM+12).

Sentence of fixed-point Data Words with 64 bit divisor: PSI_ST_TM_WDFIPOD64

This type identifies a sentence consisting of Words of type PSI_Word_Datum_Fixedpoint_D64.

It’s value is (PSI_ST_BASE_TM+13).

Sentence of one opaque Data Word with variable length: PSI_ST_TM_WDO

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_Data_Opaque.
It’s value is (PSI_ST_BASE_TM+14).

Sentence of one text Data Word with variable length: PSI_ST_TM_WDT

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_Data_Text.
It’s value is (PSI_ST_BASE_TM+15).

Sentence of one MIME Data Word with variable length: PSI_ST_TM_WDM

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_Data_MIME.
It’s value is (PSI_ST_BASE_TM+16).

Sentence of time Data Words: PSI_ST_TM_W_T

This type identifies a sentence consisting of Words of type PSI_Word_Data_Time.
It’s value is (PSI_ST_BASE_TM+17).

Sentence of one Safety Sequence Word of variable length: PSI_ST_TM_WSS

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_Safety_Sequence.
It’s value is (PSI_ST_BASE_TM+18).

Sentence of Channel Numbers: PSI_ST_TM_WCN

This type identifies a sentence consisting of Words of type PSI_Word_Channel_Number.
It’s value is (PSI_ST_BASE_TM+19).

Sentence of generic Channel Specification Words: PSI_ST_TM_W_C_S

This type identifies a sentence consisting of Words of type PSI_Word_Channel_Specification.
It's value is (PSI_ST_BASE_TM+20).

Sentence of channel status Words: PSI_ST_TM_WCS

This type identifies a sentence consisting of Words of type PSI_Word_Channel_Status.
It's value is (PSI_ST_BASE_TM+21).

Sentence of one plain opaque Word with variable length: PSI_ST_TM_WOP

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_Opaque_Plain.
It's value is (PSI_ST_BASE_TM+22).

Sentence of one plain text Word with variable length: PSI_ST_TM_WTP

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_Text_Plain.
It's value is (PSI_ST_BASE_TM+23).

Sentence of one plain MIME Word with variable length: PSI_ST_TM_WMP

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_MIME_Plain.
It's value is (PSI_ST_BASE_TM+24).

Sentence of plain time Words: PSI_ST_TM_W_T_P

This type identifies a sentence consisting of Words of type PSI_Word_Time_Plain.
It's value is (PSI_ST_BASE_TM+25).

Sentence of generic 32 bit Words: PSI_ST_TM_WG32

This type identifies a sentence consisting of Words of type PSI_Word_Generic_32.
It's value is (PSI_ST_BASE_TM+26).

Sentence of Channel Counts: PSI_ST_TM_WCC

This type identifies a sentence consisting of Words of type PSI_Word_Channel_Count.
It's value is (PSI_ST_BASE_TM+27).

Sentence of generic Node Specification Words: PSI_ST_TM_W_N_S

This type identifies a sentence consisting of Words of type PSI_Word_Node_Specification.
It's value is (PSI_ST_BASE_TM+28).

Sentence of one Reactor status Word of variable length: PSI_ST_TM_WRS

This type identifies a sentence consisting of at most one Word with variable length of type PSI_Word_Reactor_Status.
It's value is (PSI_ST_BASE_TM+29).

Sentence without content: PSI_ST_TM_WN

This type identifies a sentence containing no Words. It's main purpose is for debugging.
It's value is (PSI_ST_BASE_TM+30).

3.7.8. Detailed Reactor Status

This section describes the detailed status as sent via PSI_Word_Reactor_Status by the Reactor.

Reactor status OK: PSI_RS_OK

This constant means that there are no issues. It must not be combined with others.
It's value is 0.

Reactor uses Safe Values due to lack of data: PSI_RS_ND_SAFE

This constant means that the PSI Master has not yet sent a value for all channels, so the Reactor still uses Safe Values for these channels. All other channels are already using the Current Values.
It's value is 1.

Reactor uses Safe Values due to a timeout: PSI_RS_TO_SAFE

This constant means that no values have been received for too long, resulting in a timeout: the Reactor has switched to Safe Values. The PSI Master must send PSI_NO_FM_GODATA to the Reactor to change this status.
It's value is 2.

Reactor uses Safe Values due to an overshoot: PSI_RS_OB_SAFE

This constant means that values have been received that are outside a channel's boundaries: the Reactor has switched to Safe Values. The PSI Master must send PSI_NO_FM_GODATA to the Reactor to change this status.
It's value is 3.

Reactor uses Safe Values due to request: PSI_RS_SET_SAFE

This constant means that the PSI Master has requested the Reactor to switch it's channels to Safe Values (see PSI_NO_FM_GODATA). If no new values arrive, a timeout will occur in addition. The PSI Master must send PSI_NO_FM_GODATA to change this status.
It's value is 4.

Software error occurred: PSI_RS_SWFAIL

This constant means that the Reactor has encountered a software problem that may affect it's functionality. This does not include problems in channels.
It's value is 5.

Hardware error occurred: PSI_RS_HWFAIL

This constant means that the Reactor has encountered a hardware problem that may affect it's functionality. This does not include problems in channels.
It's value is 6.

Problem with some channels: PSI_RS_CHANNELS

This constant means that one or more channel is faulty. The PSI Master needs to query each channel's status to see which channels report which problems.

It’s value is 7.

Overheat condition: PSI_RS_OVERHEAT

This constant means that the Reactor has overheated and reduced function as much as possible. If the condition is allowed to persist, the Reactor MAY optionally develop hardware damage or switch off, so communication may cease at any time.
It’s value is 8.

PSI Master uses incorrect data types: PSI_RS_REREAD_DT

At least one channel is being fed using an incorrect data type so the data cannot be used. The PSI Master must correct the problem, possibly requiring (re-)reading of the data types.
It’s value is 9.

Reactor configuration must be re-read: PSI_RS_REREAD_ALL

This constant means that changes have been made to the Reactor that may have invalidated information previously queried, or the PSI Master seems to be using incorrect information, like sending data to invalid channels. Therefore the Reactor has switched to Safe Values, and the PSI Master must re-read all information. This may occur, for example, when channels are added, removed or swapped. As long as the PSI Master has not queried the data anew, no data are accepted.
It’s value is 10.

Reactor is ignoring commands and data: PSI_RS_MASTER_IGNORED

This constant indicates that this Reactor currently does not respond to commands sent by the querying PSI Master, and also does not apply data sent by it. This happens whenever more than one PSI Master exists (see Section 2.7), but the Reactor is configured to react to less than all of them, or to ignore that specific PSI Master. If the Reactor is configured to ignore the PSI Master but can hold it in it’s list, it responds normally in all other respects so the PSI Master will still be able to display information about the Reactor. If the Reactor’s list does not have sufficient spaces to store the presence of this PSI Master, the Reactor will not send anything except when queried for status, which will contain this flag additionally to the status.
It’s value is "bit 11".

3.7.9. Channel status

This section describes channel status values. They are binary constants intended to be put into the channel status bit field of the Word PSI_Word_Channel_Status. If the status of a channel changes to anything non-OK, then the Reactor status must reflect this by adding PSI_RS_CHANNELS. Also, if there is a change, the flag PSI_NO_TM_SUNCH must be cleared.

Channel status OK: PSI_CS_OK

If this flag is set, then the channel works normally.
It’s value is "bit 15".

The channel uses Safe Value due to lack of data: PSI_CS_ND_SAFE

If this flag is set, then the channel has not yet been fed a value and therefore is using it’s Save Value.
It’s value is "bit 14".

The channel uses Safe Value due to timeout: PSI_CS_TO_SAFE

If this flag is set, then the channel has not been fed data for a while, so a timeout occurred and the channel has switched to it’s Safe Value.
It’s value is "bit 13".

The channel uses Safe Value due to value overshoot: PSI_CS_OB_SAFE

If this flag is set, then the channel has been fed values outside it’s boundaries, so it has switched to it’s Safe Value. It’s value is "bit 12".

The channel uses Safe Value due to request: PSI_CS_SET_SAFE

If this flag is set, then the channel has been switched to it’s Safe Value. It’s value is "bit 11".

Software error: PSI_CS_SWFAIL

If this flag is set, then the channel is faulty due to a software problem.
It’s value is "bit 10".

Hardware error: PSI_CS_HWFAIL

If this flag is set, then the channel is faulty due to a hardware problem.
It’s value is "bit 9".

Overheat condition: PSI_CS_OVERHEAT

If this flag is set, then the channel has overheated and reduced functionality as much as possible. If the condition persists, the channel may develop hardware failure.
It’s value is "bit 8".

Channel does not exist: PSI_CS_VIRTUAL

If this flag is set, then the channel does not physically exist. This is no error: it merely indicates that no physical channel has been assigned to a logical one.
It’s value is "bit 7".

Channel status unknown: PSI_CS_UNKNOWN

If this flag is set, then the channel hardware has no way of checking it’s functioning, so it’s status is not known. If this is the case, PSI_CS_OK MUST NOT be returned.
It’s value is "bit 6".

3.7.10. Reactor types

This section describes the constants for the different Reactor types.

Output Reactor: PSI_RT_OUTPUT

This constant means that the Reactor possesses only Output channels, not generating any data. This is the most common Reactor type like dimmers single spots LASER systems, pyro effects, etc.. Output channels always allow reading of their Current Values.
It’s value is 0.

Input Reactor: PSI_RT_INPUT

This constant means that the Reactor possesses only Input channels, generating data but not consuming any. Reactors of this

kind include touch boards, fader panels, etc..
It’s value is 1.

Combined Input and Output Reactor: PSI_RT_INOUT

This constant means that the Reactor possesses both Input and Output channels. Reactors of this type may be non-transparent bidirectional protocol converters, sensor/actor combinations, etc.. Output Reactors always allow reading back their values, so they do not belong to this type. Reactors having only channels of type PSI_CT_NONE also do not belong to this type.
It’s value is 2.

Reactor without function: PSI_RT_NONE

This constant means that the Reactor possesses only channels of type PSI_CT_NONE or no channels at all. Reactors of this type may be testing equipment like protocol analyzers that are able to connect to a PSI Master but do not generate or consume channel data.
It’s value is 3.

3.7.11. Channel types

This section describes the constants representing the channel types.

Output channel: PSI_CT_OUTPUT

This constant means that the channel consumes data and outputs them in some way. Most channels are of this type, like dimmer channels or control channels of foggers. Channels of this type always allow reading out their Current Values.
It’s value is 0.

Input channel: PSI_CT_INPUT

This constant means that the channel creates data, for example in a fader panel.
It’s value is 1.

Combined Input- and Output channel: PSI_CT_INOUT

This constant means that the channel both creates and consumes data. Because it also outputs values, it MUST have a Safe Value

that can be read by the PSI Master in the usual way. Because pure Output channels always allow reading out their Current Values, they do not belong to this type.
It’s value is 2.

Channel without function: PSI_CT_NONE

This constant means that the channel has no function, and can neither consume nor create data. This type may be used for testing; logical channels that are not assigned to a physical channels may also have this type. Data sent to channels of this kind will be discarded, queries for values, etc. get ignored.
It’s value is 3.

Channel using Safety Sequence: PSI_CT_SAFETY

This constant means that the channel controls something that requires extra safety measures; erroneous changes must be avoided at nearly all costs. This may be a pyro effect, a LASER controller, high-voltage effects, motors, etc.. It requires correct reception of a sequence of 32 bit values to accept data. The sequence required may be queried using PSI_SO_FM_SSREQ. Any Reactor containing channels of this type MUST ensure safety if at all possible whenever it receives a query about it’s properties (like data type and even vendor information), because that indicates that the PSI Master does not (anymore) know how to access that channel, and therefore whatever value it currently uses may be invalid and thus potentially dangerous.
It’s value is 4.

3.8. Dynamics

This section describes the dynamical behavior of PSI. Only those components relevant to the discussion are looked at; for a complete list, refer to Section 3.

Note that if a message requests information, the reply must be sent as soon as possible: the node should not wait for possible further requests that might allow conglomeration of replies. However, if there already are replies that have not yet been sent, then the node may do such conglomeration. In any case, the replies generated last MUST be placed behind the previous ones, so that the sequence will always be retained. This is especially important if for some reason values for the same channel are put into the same message, since otherwise outdated values would be mistaken for the most recent ones. If the same request is received more than once, and the previous

requests have not yet been processed, then only the most recent one needs to be processed. However, a node must not wait to see if there may be duplicate requests.

The periodic status inquiries are an exception to the above: the PSI Master may delay sending of these requests for some tenths of seconds if no other messages, to which the requests could be added, are ready to be sent. However, the Reactors must still reply immediately.

3.8.1. Discovery

The use of Reactor-specific channel numbers and the resulting explicit addressing of individual Reactors necessitates that the PSI Master be informed about every single Reactor in the Nexus. Since, however, the Reactors also do not know about the PSI Master, neither side can contact the other directly to create that knowledge. Because of that, the multicast mechanism is used. All multicast traffic (including Groups (Section 2.1) and Clusters (Section 2.2)) is sent to a single multicast group: for IPv6, this is the lowest unicast-prefix based multicast address available (see [RFC3306]). For IPv4, this cannot be used, because [RFC6034] explicitly forbids the use of private IP address ranges for this use, so instead the general multicast group 225.0.0.0 is used for IPv4.

A PSI Master will continuously send messages of type PSI_MT_FM_DISCOVERY to that multicast group. Between each Discovery message sent, there MUST be a delay. The delay should default to between 1 to 5 seconds, but implementations MAY allow the user to change the setting: for a slow network, the delay may be increased, for example.
Through reception of a Discovery message from the PSI Master, Reactors are informed of the existence of the PSI Master, and are told it’s IN. Thus the Reactors can identify the specific PSI Master in case there are more than one; because the PSI Master’s IP address is delivered as well, it can be used for subsequent messages.

Only after receiving such a Discovery message from a PSI Master a Reactor may send a Discovery, under the following conditions:

- there was no contact to that Master yet (new Master) or
- that Master did not reply (REACTOR_ACCEPTED) yet or
- contact to that Master has been lost (Timeout); received Discoveries do not prevent the timeout, but periodic status inquiries do. This is MUST NOT be same timeout as for switching to Safe Values and the user MUST be able to set it independently.

If any of these conditions is met, the Reactor MUST send a single message of type PSI_MT_TM_DISCOVERY. This message is sent to the specific PSI Master as normal unicast. Under no circumstances will a Reactor send discovery messages as anything but unicast.

Whenever a PSI Master receives a Discovery message from a Reactor, it adds the Reactor to it's Nexus and acknowledges it by sending it a message containing at least one Node Section that has the flag PSI_NO_FM_REACTOR_ACCEPTED set (this MUST be set in the first Node Section for the respective Reactor). By using the multicast type (sent to the same multicast group as the Discovery messages), multiple Reactors MAY be acknowledged using a single message. Additionally, all elements that are available during normal operation may be used as well, so the acknowledgement message can be used to request information or to send data. Because the PSI Master needs to inform the new Reactor of it's maximum message length (using PSI_SO_FM_MMLINFO) before requesting data from the Reactor, this message lends itself well to doing so.
It must be noted that immediately following a (re)start of a PSI Master, messages may get dropped, because the remaining Reactors that have not yet been acknowledged are still responding to all Discovery messages. Therefore, an implementation may decide to only request information after most Reactors have already been acknowledged, defined by whatever system the implementor deems reasonable. Because Reactors never send Discoveries by themselves, a PSI Master MAY decide to temporarily stop sending Discovery messages until it has sent acknowledgements to all Reactors it received Discoveries from, but MUST resume sending normally when it has done so.
Network effects can make the PSI Master receive a Reactor's Discovery message after the Reactor has received an acknowledgement by the PSI Master. Because the PSI Master can not know if the Reactor actually received the acknowledge or not, it must send an acknowledge whenever it receives a Discovery message. The Reactor must therefore be prepared to handle this, for example by discarding previous or excess acknowledgements.

The result is "at least once" semantics. Every message is idempotent, so that the entire process can be restarted by either side at any time, while the number of (identical) messages received by either side has no influence on the outcome.

Whenever a Reactor has not received any messages from the PSI Master (Discoveries and messages not addressing the specific Reactor do not count towards this, but periodic status inquiries addressed at the Reactor do), it MUST time out and respond to the next Discovery message from that PSI Master by sending a unicast Discovery message to it. The timeout SHOULD be user-configurable and default to at least 30 seconds and at most two minutes. It is different from the

timeouts used for Safe Values.

This way, all nodes can be relatively certain that, in case one gets restarted, the respective partner knows that the connection needs to be recreated. It cannot be 100% reliably be ensured if one takes into account that restarts may get masked by certain combinations of cabling disconnects and duplicated, old packets, but even then there is no real problem, because the IN is unique. If two or more Reactors receive swapped IP addresses after a restart, and the PSI Master erroneously sends messages to the wrong Reactor, this will be noticed by the receiving Reactor because it's IN does not match the TIN from the message. In the simplest case, it will just ignore the entire message, so the PSI Master will not receive any replies and regard the Reactor as lost, as well as the Reactor timing that PSI Master out. In any case it is ensured that no incorrect data are used.

Reactors do not send a reply after receiving the acknowledgement from the PSI Master; they just cease sending Discovery messages. This means that the PSI Master does not know if the Reactor has received the acknowledgement, or if it was disconnected or has failed. However, this information is automatically gathered by way of the periodic status inquiries (see Section 3.8.3).

Because a Reactor might erroneously keep sending Discovery messages even after receiving acknowledgements, the PSI Master SHOULD implement a user-configurable maximum of ignored acknowledgements, and should communicate the details to the user so that the Reactor can be inspected.

3.8.2. Initialization

After Discovery, Initialization occurs. It is used to query information about the Reactor that has been discovered.
Every Reactor is characterized by a number of properties describing it's function and uses. The PSI Master sends specific queries to the Reactor to collect this information. All such information may be queried in a single message, or distributed across several messages, possibly containing data sent to the Reactor. The sequence of the queries may be chosen arbitrarily, as well as whether to wait for each reply before sending the next query or not. Also, as described in Section 3.7, some need not be queried in all cases. The individual queries are:

1. Reactor type

2. Reactor status

3. vendor specific information

4. user specific information

5. channel counts

6. channel types

7. data formats

8. data boundaries

9. maximum message length

10. maximum refresh rate

11. channel status

12. GID assignments

Especially with Reactors having many channels, replies to one or more queries may need to be split into multiple messages. It is within the Reactor's discretion to decide in which way to do this, so parts of different replies may be conglomerated into a single message. The PSI Master must thus be prepared to re-request any part of the required information that may have been lost in transit.

Reactors MUST reply to all of these requests, but may decide whether to combine multiple replies or not.

When all information is known to the PSI Master, Initialization is complete.

3.8.3. Normal Operation

Periodic status inquiries In order to always be informed about the status of the Reactors in the Nexus, the PSI Master must query their status periodically. The interval SHOULD be configurable, but SHOULD NOT be made smaller than about 1 Hz, because the information is evaluated by the user, resulting in much larger latency. This way, the network is not overly burdened by these inquiries. Since these queries can easily be added to messages that need to be sent anyway, the overhead can be further reduced. Since nearly all Reactors are expected to receive data or queries at a much smaller interval, almost no extra messages will need to

be sent to accommodate this. On the other hand, a Reactor MUST NOT delay replies to status inquiries in order to combine them with another message, but MAY do so, within length limits, if there is a message pending for sending already. This may be convenient if pre-arbitration (not discussed in this memo) is employed to schedule messages sent to the PSI Master. If no message is pending for sending, a dedicated message must be sent.

Data messages This is the most common form of communication, as data messages are the core intention of this protocol. The goal is to reach the highest refresh rate possible, so data must be sent as soon as possible. Especially, a PSI Master must not wait for data to be generated before sending what has already been generated. This does however not mean that a message must be sent for each channel of a Reactor. Actually, a PSI Master must strive to minimize the number of messages sent by packing data for as many channels as possible into every message, for example by arranging data generation or consumption in a way that ensures data can be sent and received at any time.
However, the maximum message length may be dynamically adjusted to reach the lowest net data loss if the network proves unreliable. In that case, the PSI Master may distribute data for any Reactor's channels across multiple messages, but keeping identical data types logically ordered is still sensible. This is because using a message of only a single Word type saves the overhead of requiring multiple Sentence headers, reducing message size and therefore probability for errors, as well as bandwidth needed. In such a case, multicast messages also might deliberately contain data for fewer Reactors than it normally would, in favor of fewer errors. Loss feedback methods are not discussed as part of this memo, however. An implementation MAY allow the user to set a maximum message length that best suits their network.
A data message itself does not require a reply, but may be combined with requests.

Data requests If no pre-arbitration (not discussed in this memo) is performed, data generated by Input and InOut Reactors must be explicitly requested by the PSI Master (polling). As before, the objective is a refresh rate that is as high as possible.
Therefore it is imperative to request data from all channels of any specific Reactor at once. This way ensures that network latency is experienced only once per direction and Reactor. As with data messages, an unreliable network may make splitting requests, and subsequently, replies, into multiple messages in favor of smaller size preferable.
Data requests MUST always be replied to immediately, but MAY be

combined with other messages if such messages are about to be sent already. Waiting for additional requests is not allowed. An implementation MAY allow the user to set a maximum message length that best suits their network.

Information messages An information message includes anything that are not channel data, like data types or GID assignments. They are therefore not merely "informational", as the name might suggest. Information messages usually appear during initialization of the Nexus, and therefore their impact on network performance is small during normal operation. They do not require acknowledgement, but may be combined with requests. They must be sent immediately, waiting for more information to be requested is not allowed.

Information requests This includes anything that is neither a data request nor a periodic status inquiry, like requests for data types, data boundaries or Reactor type. They appear mostly during Nexus initialization, which the main cause of information messages, rendering their influence on network performance small. They may be combined with other information or data messages, but replies still MUST be sent immediately. Therefore, it is advisable to send all information requests for any specific Reactor that will only generate minimal reply data within a single message, to reduce the number of messages that have to be sent. Things like Reactor type, Reactor status, channel counts and maximum message length are good candidates for this. However, if the network is very unreliable, a higher number of smaller messages may be preferable even for this kind of messages.

Lost nodes Whenever a node does not receive replies (like periodic status inquiries) after 40 attempts, then the node is considered to be "lost". This will happen if the node has failed, is turned off or disconnected from the network. This applies to both Reactors and PSI Masters.

Timeouts Whenever a Reactor does not receive data messages from the PSI Master, it MUST switch all channels to Safe Values. The time used for the timeout MUST be chosen according to the potential harm caused by the most dangerous channel of the Reactor, but SHOULD NOT default to than ten seconds in any case. For dimmers, for example, this timeout is not critical, as they might cause fire only after some minutes, if at all. A LASER unit, especially if it does audience scanning, is much more critical, so the

timeout time must follow local safety regulations unless other measures are put into place. This value therefore MUST be user configurable.
A second timeout is used for values received using ACKed mode, because these values are usually intended to stay much longer. This value, too, MUST be user configurable, and the user may disable it entirely. It should default to one minute.
Additionally, for ACKed mode, the user may select that the timeout does not time out when status inquiries are received, and SHOULD also be able to disable the timeout entirely.

If a single channel doesn't receive values anymore, then it, too, must switch to it's Safe Value just like an entire Reactor. This also applies to channels using the Safety Sequence, that MUST be cancelled upon timeout.

Like any other channel, a channel using Safety Sequence times out if it does not receive new values. However, contrary to any other channel, this status is normal for an inactive channel using Safety Sequence: data for such channels needs to only be sent when the channel's value is supposed to change to or remain as anything possibly non-safe. Upon timeout, such a channel MUST therefore immediately cease operation and ensure safety if at all possible. Only correctly received and matching parts of the Safety Sequence keep it from timing out.

If a PSI Master doesn't receive values from an Input or InOut channel or Reactor anymore, then it SHOULD set all channels that are controlled partially or in full by those channels to use Safe Values, especially if they use the Safety Sequence. In any case, use of Input or InOut channels to control safety-critical channels is not recommended, as values may be kept longer than they are valid. The times and configuration for both modes are defined just like those for Reactors.

4. Summary of operation

To provide an overview of the workings of the PSI, this section describes a minimal subset that will result in a system that will do discovery and communication between one PSI Master and any number of Reactors, using a single data type only. No advanced functionality like Groups is included, nor is handling of disconnection or lost nodes. Maximum message lengths can be hardcoded at the implementor's discretion. Features adding reliability, safety or avoidance of load peaks are mostly omitted as well, so this simplistic version will likely only work properly with a handful of connected Reactors. It is intended to be used as a starting point to become familiar with the system; all the missing features can be added later.

4.1. PSI Master

After starting up fully, the PSI Master begins sending Discovery messages to the Discovery multicast group (see Section 3.8.1). Whenever it receives a Discovery message from a Reactor, it looks up the Reactor in it’s list, and adds it’s IN and IP address if it does not yet have an entry. In any case, it marks the Reactor as "new", and acknowledges it by sending it a message with the PSI_NO_FM_REACTOR_ACCEPTED flag set in the node header (see Section 3.5.2). That message need not contain any further content, so no sentence header exists.
Next, the PSI Master sends a message querying the Reactor for it’s Reactor type and channel counts (see Section 3.8.2), using the flags PSI_NO_FM_RTREQ and PSI_NO_FM_CCREQ in the node header. After receiving the reply, the PSI Master queries the Reactor for it’s channel types and data types, using PSI_NO_FM_CTREQ and PSI_NO_FM_DTREQ, respectively.
When these are known, the Reactor is initialized and can be used. In Redundancy Mode, the PSI Master will send several messages per second to the Reactor, containing the data it is supposed to use. The sentence type to be used is the one that was received in reply to the PSI_NO_FM_DTREQ flag; because the example Reactor (see below) expects PSI_Word_DatumU8, that will be PSI_ST_FM_WDU8. To tell the Reactor that it new values shall be set, the PSI Master sets the PSI_SO_FM_VSET flag in the sentence header (see Section 3.5.3).

Note that the PSI Master will never cease sending Discovery messages, so Reactors can be added at any time and immediately become usable without user interaction.

4.2. Reactor

Upon startup, a Reactor joins the Discovery multicast group (see Section 3.8.1), so that it can keep track of all PSI Masters within the Nexus. Whenever it receives a Discovery message, it will look up the sending PSI Master in this list. If the PSI Master is not in the list, it’s IP address and IN are added the PSI Master is marked as new. In this case, or if the PSI Master already is in the list and it’s status is "new", the Reactor will send a single Discovery message in reply. If the PSI Master’s status is not "new", then no action will be taken.
When the Reactor receives an acknowledgement message, it looks up the sending PSI Master in it’s list and changes it’s status to "acknowledged". As mentioned above, it will not react to Discovery messages from that PSI Master afterwards.
The Reactor will then receive queries for Reactor type and channel counts (see Section 3.8.2), and then for it’s channel types and data formats. The most common Reactor type, Output, will send a sentence

of type PSI_ST_TM_W_N_S with the flag PSI_SO_TM_RTINFO set in the sentence header (see Section 3.5.3), containing a single Word of type PSI_Word_Node_Specification with the "node specification" set to PSI_RT_OUTPUT. On the other hand, a sentence of type PSI_ST_TM_WCC with a single word of type PSI_Word_Channel_Count conveys the channel counts (one Output, no InOut and no Input for example).
In reply to the channel type request, a sentence of type PSI_ST_TM_W_C_S with the flag PSI_SO_TM_CTINFO set is sent that contains as many words of type PSI_Word_Channel_Specification as there are channels. In this example, there will be a single word, containing channel number 0 and the "channel specification" set to PSI_CT_OUTPUT. Because there only is a single channel, 8 bit suffice for the channel number size, so the flag PSI_SO_TM_CN8 is set. Setting the proper flag for the channel number size is required so that the receiver can correctly calculate the number of words from the sentence length.
Finally, in reply to the request for the data types, the Reactor sends a sentence of type PSI_ST_TM_W_C_S with the flag PSI_SO_TM_DTINFO set. The example Reactor’s Output channel expects to be fed PSI_Word_DatumU8 from the master, so it sets the "channel specification" to PSI_ST_FM_WDU8.

5. Routing Considerations

In normal environments, the presence of routers is neither required nor recommended. If routers are employed, this protocol’s reliance on multicasts for specific functions must be kept in mind.
In any case, the PSI is not intended to be used directly on the internet, as it lacks congestion control and similar features.

6. Security Considerations

This protocol is intended for use in a tightly controlled environment without publicly accessible connection points only. If a connection to external networks like the Internet exists, it is expected to be protected by appropriate measures like firewalling, that block all PSI related traffic in both directions. This assumption is reasonable given that the control wiring (rigging) is done outside of the audience’s reach, and it may be assumed that the personnel with access to the rigging is reasonably trusted. Likewise, operations using this type of installation have no need for external connectivity on the production devices.
Similarly, the environment described is a low-risk target that would neither yield sensitive information when penetrated nor allow significant damage to be dealt when taken over or sabotaged.
Additionally, implementing security measures, even as simple as plain text passwords, would impair quick installation, exchanging of devices and automatic discovery, that are central features of the

PSI. The cost of adding such security measures would therefore be unacceptably high compared to the rather low risk of attacks. Therefore, no provisions are made for confidentiality, authentication or other security measures. An attacker might overflow the Reactor's list of PSI Masters by posing as more PSI Masters than it can hold. This requires reception of the Discovery messages however, so due to the routing restrictions listed in Section 5 the attacker would normally need to be situated on the same network.
Should this protocol be required to span an unsafe network, existing security technologies like IPSec must be used. Depending on the actual security technology employed, the routing considerations in Section 5 may or may not apply in this case.

While the Safety Sequence capability of the PSI requires an attacker to be able to receive messages in order to maliciously activate the devices, it is not intended to safeguard against attacks; it is meant to protect against transmission errors, hardware or software failure, and misconfiguration. It is further noted that current controllers for this application may, in theory, be compromised as well, though the general availability of compatible networking devices, that is one of the strengths of this protocol, and the expected proliferation of unobserved connection points creates a much easier target than the comparatively direct wiring of a conventional controller.

In addition, the MIME handling capabilities of a PSI implementation are in theory exposed to the same risks as an email client handling the same types. An attacker could send messages containing fake VSI replies containing malicious MIME content, if they know the IN and address of at least one node. However, given that disruption can likely be achieved more easily, and also that MIME capabilities will be much more limited in a PSI implementation, and finally that the potential gain of compromising even the PSI Master is rather low, this does not seem to be a likely occurrence.

7. IANA Considerations

This protocol requires the assignment of two port numbers: one for the default PSI Master discovery reception port and one for the Reactor discovery reception port. Two distinct ports are necessary so that both PSI Master and Reactor may run as separate applications on the same IP address.

The ports assigned must be above 1023 so that the PSI does not require special privileges to be used. While only UDP is used on the Reactor side, the corresponding TCP port might be assigned as well so that the management / control protocol that may be created at a later date will not require a new assignment. The inter-master communication runs across TCP so this port is required. Currently,

ports 7911 (short name: PSI\Reactor, long name: Protocol for Stage Illumination (Reactor)) and 4919 (short name: PSI\Master, long name: Protocol for Stage Illumination (Master)) are used.

Additionally, for IPv6, the lowest-numbered unicast-prefix-based multicast group as per RFC3306 is used for discoveries, group and cluster messages, while with IPv4, the multicast group 225.0.0.0 is used for that purpose. It may be desirable to assign dedicated multicast groups instead.

8. References

8.1. Normative References

[CORBA-CDR] Object Management Group, OMG., "Common Object Request Broker Architecture (CORBA) Specification, Version 3.1 - Part 2: CORBA Interoperability, section 9.3: CDR Transfer Syntax", January 2008.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, August 2002.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

8.2. Informative References

[DMX512-A] ESTA, ESTA., "Entertainment Technology - USITT DMX512-A - Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories", 2004.

[RFC4506] Eisler, M., "XDR: External Data Representation Standard", STD 67, RFC 4506, May 2006.

[RFC6034] Thaler, D., "Unicast-Prefix-Based IPv4 Multicast Addresses", RFC 6034, October 2010.

[TAO] Schmidt, D., "The Ace Orb Home Page", June 2007, <http://www.cs.wustl.edu/~schmidt/TAO.html>.

Author's Address

Marcus Ertl
An Peschbenden 13
47906 Kempen, NRW
DE

EMail: marcus.ertl@gmx.net