Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 29, 2014

M. Ertl
May 28, 2014

Protocol for Stage Illumination
draft-mertl-psi-04

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.

Due to the complexity and length, versions 4 and above only describe a subset of the full PSI.

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. Please note that, starting with this version, only a very basic version of the PSI is actually described. This simple version lacks many important features and will thus not conform entirely to the full version. The intent is to provide a starting point for implementation that can be built upon and converted to the full version with relative ease.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD 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 5.3). 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 data types 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 5.4). 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 4.1 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.

3. 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 hard coded at the implementer'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.

3.1. PSI Master

After starting up fully, the PSI Master begins sending Discovery messages to the Discovery multicast group (see Section 4.1).
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 5.5.2). 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 4.2), 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 5.5.3).

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.

3.2. Reactor

Upon startup, a Reactor joins the Discovery multicast group (see Section 4.1), 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 4.2), 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 5.5.3), 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.

4. 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 5.

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.

4.1. 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 (see [RFC5771]) 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:

- o there was no contact to that Master yet (new Master) or

- o that Master did not reply (REACTOR_ACCEPTED) yet or

- o contact to that Master has been lost (Timeout); received Discoveries do not prevent the timeout, but periodic status inquiries do. Because the Master may have stopped sending data but is still querying the status, the Reactor would switch to Safe Values but not regard the Master as lost. Therefore, and because the safety timeout normally is shorter than the Master timeout, there must be separate timeouts, and the user MUST be able to set both 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 implementer 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 4.3).

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.

4.2. 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 5.7, some need not be queried in all cases. The individual queries are:

1. Reactor type
2. Reactor status
3. vendor specific information
4. channel counts
5. channel types
6. data formats
7. data boundaries
8. maximum message length
9. maximum for GID assignments
10. channel status
11. 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.

4.3. 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.

Metadata messages A metadata message includes anything that are not channel data, like data types or GID assignments. These 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.

Metadata requests A metadata request includes anything that is neither a data request nor a periodic status inquiry, like requests for data types, data boundaries or Reactor type. These 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.

5. Design

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

5.1. System-independent representation (transfer syntax)

In order to not add yet another protocol-specific data representation that would require implementers 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.

5.2. 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 5.7.4) 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.

5.3. 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 5.1 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.

5.4. 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:

o 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 5.7).

o 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 5.7).

o 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 4 for details).
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.

5.5. Headers

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

5.5.1. 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 (broadcast SHOULD NOT be used), 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 ID (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.
Messages of type PSI_MT_xx_UNICAST SHOULD NOT be sent as multicast or broadcast.

Over the wire format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |D+E+S+  Type   |         Message Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| : . Message Length (ctd.) . : |              SIN              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           SIN (ctd.)                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          SIN (ctd.)           | : . : . : . :GID: . : . : . : |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| : . : . :GID (ctd.) : . : . : |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 5.2 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 5.4 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 5.7.3), 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 5.3). 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.

5.5.2. 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 5.5.3).

Over the wire format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Node Options                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Section Length        + : . Section Length (ctd.) . : |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| : . : . : . : . : . : . : . :TIN: . : . : . : . : . : . : . : |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| : . : . : . : . : . : . : . :TIN (ctd.) : . : . : . : . : . : |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| : . : . : Request ID: . : . : +
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Legend:
: . XYZ . : = octets that only exist when certain conditions are met
```

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 5.7.5.

Section Length: 16 / 32 bit

Similar to the length field of the message header (Section 5.5.1), 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 5.3). 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 5.7.5). 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.

5.5.3. 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:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Sentence Type |       Sentence Options        |Sentence Length|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S. Len. (ctd.) + : .Sentence Length (ctd.) . : |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 5.6 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.

5.6. 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 5.1) 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.

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".

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".

Data Word for plain text: PSI_Word_Text_Plain

This is a 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: "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.

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".

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 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".

5.7. Constants

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

5.7.1. Conventions

The conventions used in this document are as follows:

o 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".

o 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".

5.7.2. Protocol version

The currently existing versions are:

PSI 32 Bit: PSI_PV_0

This is the full protocol described prior to this document.
It is defined as 0.

Simple PSI 32 Bit: PSI_PV_S

This is the simple protocol described here.
It is defined as 1.

5.7.3. 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.

5.7.4. Message types

The following constants identify the message types as described in Section 5.4. 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 5.5.1, 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).

5.7.5. Node Options

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

5.7.5.1. 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 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), if any. If nothing has been set, an empty sentence of the respective type is sent.
It’s value is "bit 21".

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 5.5.1), 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".

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 4 for details.

5.7.5.2. 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".

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".

5.7.6. 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.

5.7.6.1. 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".

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 5.7.5). 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 5.7.5). 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 5.7.5). 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".

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".

5.7.6.2. 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".

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".

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 PSI_ST_TM_WTP.
It’s value is "bit 3".

5.7.7. Sentence types

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

5.7.7.1. Sentence types for messages from a PSI Master

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 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 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 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).

5.7.7.2. Sentence types for messages to a PSI Master

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 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 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 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 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).

5.7.8. 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.

5.7.9. 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".

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".

5.7.10. 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, pyrotechnic devices, 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.

5.7.11. 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 fog machines. 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.

6. Routing Considerations

In normal environments, the presence of routers is neither required nor recommended. If routers are employed, this protocol’s reliance on multicasting 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.

7. 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 firewalls 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.

8. 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 (listed as "RESERVED" in RFC5771) is used for that purpose. It may be desirable to assign dedicated multicast groups instead.

9. References

9.1. 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.

[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.

9.2. 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.

[RFC5771] Cotton, M., Vegoda, L., and D. Meyer, "IANA Guidelines for IPv4 Multicast Address Assignments", BCP 51, RFC 5771, March 2010.

[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