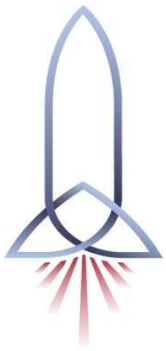
School of Computing
DUNDEE

SPACE SYSTEMS RESEARCH GROUP



Space
Technology
Centre
University of Dundee

SpaceWire-PnP

Protocol Definition

Revision: Draft A Issue 2.1

Date: 16th September 2009

Ref: SpW-PnP-PD

Space Technology Centre
School of Computing
University of Dundee
Dundee, DD1 4HN
Scotland, UK
spacetech.computing.dundee.ac.uk

Document Authors

Peter Mendham

Albert Ferrer Florit

Steve Parkes

Document Change Log

Date	Revision No	Comments
16th September 2009	Draft A Issue 2.1	Skeletal SpW-10X mappings documented
24th February 2009	Draft A Issue 2.0	Restructure to clarify (levels introduced)
17th February 2009	Draft A Issue 1.1	Changes to data source/sink architecture
9th February 2009	Draft A Issue 1.0	First issue

A comprehensive list of the changes made to this document in each major revision is provided in section 10.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 AIMS AND OBJECTIVES

The SpaceWire standard [AD1] defines the aspects of a highly flexible and capable communication system which roughly correspond to the physical and data-link layers of the ISO Open Systems Interconnection (OSI) basic reference model [RD1]. The standard also defines a number of features which could correspond to the network layer of this model. Whilst following the standard does ensure a certain degree of interoperability, one which is further extended by the standard protocol suite and protocol identification mechanism [AD2], a SpaceWire network must still be constructed, and configured, carefully for a given application, usually requiring customised software and/or hardware. The lack of standardisation for simple tasks required on almost all SpaceWire networks limits the level of interoperability which may exist between devices and software, and the extent to which both hardware and software can be re-used between different applications.

The aim of SpaceWire-PnP (Plug-and-Play) is to provide standardised, interoperable mechanisms for performing key functions associated with SpaceWire networks. The term 'plug-and-play' originates from the commercial electronics market where a range of techniques were developed to improve the user experience of device integration. From the perspective of a user, application of the term 'plug-and-play' indicates that it should be possible to interface two or more arbitrary devices without the need for configuration. Plug-and-play generally involves two key aspects:

1. Automatic discovery and configuration of hardware and software systems in response to changes in physical interfacing or availability, including whilst the system is running ('hot-plugging').
2. Detection and configuration of the services that a plug-and-play enabled device provides.

SpaceWire does not offer a standard mechanism for detecting the topology of a network, or what devices are attached to it. Nor does it offer a standard mechanism for configuring the various aspects of a SpaceWire network, such as links and routers. SpaceWire also lacks standard features to assist detection or configuration beyond the network, in the service domain. It is the aim of the SpaceWire-PnP protocol to add these features, within the scope of what is practical.

WARNING

This current document is an early draft of the proposed standard and is for discussion purposes only. It will change after prototyping work has been completed. Applicable documents may also change.

DO NOT USE THIS DOCUMENT TO DESIGN DEVICES OR SYSTEMS!

1.2 BACKGROUND

The SpaceWire-PnP protocol defined in this document arose out of earlier work conducted by the Air Force Research Laboratory (AFRL), the Naval Research Laboratory (NRL) and NASA Goddard Space Flight Centre as part of the Operationally Responsive Space (ORS) programme. This work resulted in a SpaceWire Plug-and-Play document [RD2], defining a new protocol which performed the tasks necessary for the ORS programme at that time.

After the circulation of this document, the University of Dundee became involved with five main aims:

1. Ensuring that plug-and-play leverages existing technologies as much as possible, rather than creating unnecessary, bespoke solutions.
2. Extending the feature set of plug-and-play to ensure that it covers all aspects of SpaceWire defined in the SpaceWire standard, with additional facilities to cover the most common, practical use cases.
3. Identifying issues and correcting bugs in the mechanisms proposed by the earlier documents; clarifying many of the use cases.
4. Ensuring that plug-and-play fits within the framework of SpaceWire protocols and software frameworks current under development.
5. Providing a mechanism by which existing technologies can be incorporated into a plug-and-play system.

This work has lead to the development of the SpaceWire-PnP protocol, which is defined in this document.

1.3 GUIDE TO DOCUMENT

Section 2 lists the terms and definitions relevant to the SpaceWire-PnP protocol.

Section 3 provides an informative overview of the SpaceWire-PnP protocol, describing the approach that the protocol takes and the relationship between the various services that it provides.

Section 4 specifies normative details of the SpaceWire-PnP protocol at an overview level.

Section 5 presents a normative description of level 1 SpaceWire-PnP services specifying target parameters, target actions and initiator primitives.

Section 6 presents a normative description of the differences between level 2 and level 1 SpaceWire-PnP services specifying additional and modified target parameters, target actions and initiator primitives over and above those required for level1 support.

Section 7 presents a normative description of standard capability services for providing data sources and data sinks.

Section 8 presents a normative description of the use of the RMAP packet format and protocol semantics by SpaceWire-PnP.

Section 9 describes how the protocol may be adapted to permit the SpW-10X routing device (Atmel IC AT7910E) to be used in SpaceWire-PnP systems.

Changes to this document are described in detail in Section 10.

1.4 ACRONYMS AND ABBREVIATIONS

AD	Applicable Document
AFRL	Air Force Research Laboratory
API	Application Programming Interface
CRC	Cyclic Redundancy Code
ECSS	European Cooperation for Space Standardization
EGSE	Electrical Ground Support Equipment
GAR	Group Adaptive Routing
GSFC	Goddard Space Flight Centre
IC	Integrated Circuit
I/F	Interface
NASA	National Aeronautics and Space Administration
NRL	Naval Research Laboratory
ORS	Operationally Responsive Space
OSI	Open Systems Interconnect
PD	Packet Distribution
PDU	Protocol Data Unit
PnP	Plug-and-Play
QoS	Quality of Service
RD	Reference Document
RMAP	Remote Memory Access Protocol

RMW	Read/Modify/Write
SOIS	Spacecraft Onboard Interface Services
SpW	SpaceWire
UoD	University of Dundee

1.5 REFERENCE DOCUMENTS

The documents referenced in this document are listed in Table 1-1.

Table 1-1: Reference Documents

REF	Document Number	Document Title
RD1	ISO/IEC 7498-1:1994. 2nd ed.	Open Systems Interconnection—Basic Reference Model: The Basic Model
RD2		Original PnP draft document (January 2008)
RD3	IAC-07-B4.7.06	SpaceWire Plug-and-Play: Drivers and Alternatives (Presented at the 48th Congress of the International Astronautical Congress, Hyderabad, India)
RD4	UoD_SpW-10X_UserManual	SpW-10X SpaceWire Router User Manual (Issue 3.4)

1.6 APPLICABLE DOCUMENTS

The documents applicable to this document are listed in Table 1-2.

Table 1-2: Applicable Documents

REF	Document Number	Document Title
AD1	ECSS-E-ST-50-12C, January 2003	SpaceWire: Links, nodes, routers and networks
AD2	ECSS-E-ST-50-11C Draft 0.5	SpaceWire Protocols Feb 2008
AD3	SpW-RT WP3-200.1	SpaceNet –SpaceWire-RT: Initial Protocol Definition
AD4	-	The Unicode Standard, Version 5.0, 2007

2 TERMS AND DEFINITIONS

In this section the terms and definitions proposed for the SpaceWire-PnP standard are provided in the form required by ECSS.

2.1 DEFINITIONS FROM THE OPEN SYSTEMS INTERCONNECTION (OSI) BASIC REFERENCE MODEL

layer subdivision of the architecture, constituted by subsystems of the same rank

protocol data unit (PDU) unit of data specified in a protocol and consisting of protocol-control-information and user data.

service capability of a layer (service provider) together with the layers beneath it, which is provided to service-users.

service data unit (SDU) set of data which is semantically unchanged when transferred between peer entities in a given layer and which is not interpreted by the supporting entities in that layer.

2.2 TERMS DEFINED BY THE SPACEWIRE STANDARD

byte 8 bits, numbered 0 to7, where 7 is the most significant

control character character that is used to pass control information across a link

data character data byte encoded ready for transfer across a link

data signalling rate rate at which the bits constituting control and data characters are transferred across a link

destination node or unit that a packet is being sent to

destination address route to be taken by a packet in moving from source to destination (path address) or an identifier specifying the destination (logical address)

end of packet marker control character which indicates the end of a packet

link bidirectional connection of one unit to another unit for passing data and control information

logical address data character at the start of a packet, which identifies the destination for the packet

Mb/s 1 000 000 bits per second

network set of units connected together via links and routing switches

node source or destination of a packet, which can be a processor, memory unit, sensor, EGSE or some other unit connected to a SpaceWire network

normal-character data character or control character (end of packet marker)

path address series of one or more data characters at the start of a packet which define the route to be taken across a SpaceWire network

packet sequence of normal-characters comprising a destination address, packet cargo and an end of packet marker

packet cargo data to transfer from a source to a destination

router routing switch

routing switch switch connecting several links that routes packets from one link to another where the destination address of each packet by the switch is used to determine which link a packet is sent out on

source node or unit sending a packet

unit box, board or subsystem, that can have one or more SpaceWire interfaces

2.3 Terms Defined by the Remote Memory Access Protocol (RMAP) Standard

For the purpose of this Standard, the terms and definitions from ECSS-E-ST-50-11A apply.

2.4 Terms Defined in this Document

For the purposes of this document, the following definitions also apply. Many other terms that pertain to specific items are defined in the appropriate sections.

action the activities which a target carries out under specified conditions beyond the required activities of responding to SpaceWire-PnP commands with replies.

active node a SpaceWire-PnP node which acts as an initiator for SpaceWire-PnP commands

capability a protocol, or standardised protocol feature (such as RMAP address space), optionally transported via another protocol.

capability service a service provided to permit the interrogation and configuration of capabilities.

configuration port a destination address of zero, present on every plug-and-play device, to which plug-and-play packets should be addressed

core service the services provided by SpaceWire-PnP which relate to features of SpaceWire, rather than higher layer protocols or additional features.

device a node or routing switch

entry complex parameters consist of multiple entries: a single root entry and one or more non-root entries.

field each parameter entry is split into multiple fields which encode the information associated with that parameter and service.

managed network a SpaceWire-PnP network in which there is no competition between active nodes, any competition is resolved by design. Also referred to as a level 1 network.

non-root entry a complex parameter has zero or more non-root entries each of which have the same structure of fields.

open network a SpaceWire-PnP network which uses SpaceWire-PnP algorithms to resolve conflict between multiple active nodes and to permit the coexistence of multiple active nodes all of which are actively participating in the network. Also referred to as a level 2 network.

owner on an open network every device has an owner which is an active node responsible for managing that device. To permit other active devices to interrogate and use the device the owner must provide a proxy.

parameter related information provided by targets to the SpaceWire network. A parameter may be simple (single entry) or complex (multiple extries).

passive node a node which acts solely as a SpaceWire-PnP target for SpaceWire-PnP commands.

primitive an access point for initiators through which higher layers may access information and facilities provided by a service.

proxy a copy of all of the parameters of a device provided by the device owner on an open network to permit controlled access to that device. Each proxy is identified by a proxy ID and a proxy key.

proxy ID identifies an owner proxy on an open network.

proxy key helps with the unique identification of an owner proxy on an open network.

region a contiguous portion of a SpaceWire-PnP network which may or may not correspond to a logical address region.

root entry the single root entry of a complex parameter presents fields which are related to all parameter entries such as the number of non-root entries

service in addition to the definition above, SpaceWire-PnP services provide functionality to the layers beneath, via a SpaceWire interface. As such, a service consists of target parameters, actions and initiator primitives.

3 PROTOCOL OVERVIEW (INFORMATIVE)

In this section an informative description of the SpaceWire-PnP protocol is provided as an introduction to the subsequent normative specifications.

3.1 SCOPE

The central goal of SpaceWire-PnP is interoperability at the network level. As such SpaceWire-PnP provides services to discover, identify and configure the features of a SpaceWire network, as covered by the SpaceWire standard. SpaceWire-PnP also includes support for a number of features which correspond to the most common practical use cases of SpaceWire. SpaceWire-PnP does not require devices to support more of the SpaceWire standard than is necessary to achieve their objectives: within reason, if something is optional in the SpaceWire standard, SpaceWire-PnP does not require that it be implemented.

As SpaceWire-PnP is intended to cover all features of the SpaceWire standard, it is also limited to these features. The services provided by SpaceWire-PnP extend only as far as the network layer. It is the intention of SpaceWire-PnP to provide the necessary mechanisms to ensure a high level of interoperability. Further interoperability, such as a wide range of standardised data formats for data sources and sinks, is not considered to be within the scope of SpaceWire-PnP.

3.2 NETWORK PERSPECTIVE

SpaceWire-PnP views a SpaceWire network in a manner consistent with the SpaceWire standard. A network consists of *nodes* and *routers* (collectively known as *devices*). Each device has a number of *ports*: a node has one or more ports; a router has two or more ports. The maximum number of ports a device may have is 31. The first port on a device is always 1, and ports are numbered contiguously up to the highest port on the device. Ports may be connected via bidirectional *links*.

3.2.1 Routing Functions and the Configuration Port

Routers are expected to provide a routing function which is consistent with the SpaceWire standard, permitting the wormhole routing of packets by path addressing and, optionally, by logical addressing using an internal routing table. Nodes are expected to provide no routing function: packets arriving at any port on a node will be consumed by the node.

All SpaceWire-PnP devices (both routers and nodes) interpret packets arriving at the device with a leading zero as containing configuration information. This can be thought of each device having an internal “configuration port” to which these packets are addressed. When discovering a network, it is often not possible to know which port on the device is being used to interrogate it. In this case it is

impossible to specify the first part of a return address for a packet. To resolve this issue, SpaceWire-PnP reply packets are always sent out of the port through which the command was received, irrespective of the addresses specified for the reply.

3.2.2 Network Regions

SpaceWire-PnP views a network as being assigned to one or more *regions*. This can easily correspond to the concept of regional logical addressing, where each region is the extent to which single logical addresses are valid, with a logical address prefix used to transfer packets between regions (see the SpaceWire standard for more information [AD1]). Regions do not have to be used for this purpose in SpaceWire-PnP, and may not be used at all, but can be helpful in the management of a logical address assignment policy. Such policies are considered to be beyond the scope of SpaceWire-PnP, and are more of an issue for system designers and integrators.

3.2.3 Network Topology

In general, SpaceWire-PnP does not restrict network topology. However, the mechanisms used for the network discovery do place a restriction on how a multi-port node may be connected if a network topology is to be fully determined. If a node has multiple connected ports, then the devices that are accessible via these ports also need to be mutually accessible (either directly, or via other routers). So, for example, a multi-port node cannot be used as a bridge between two distinct networks (whatever the size), if full network discovery is expected. This is shown in Figure 3-1.

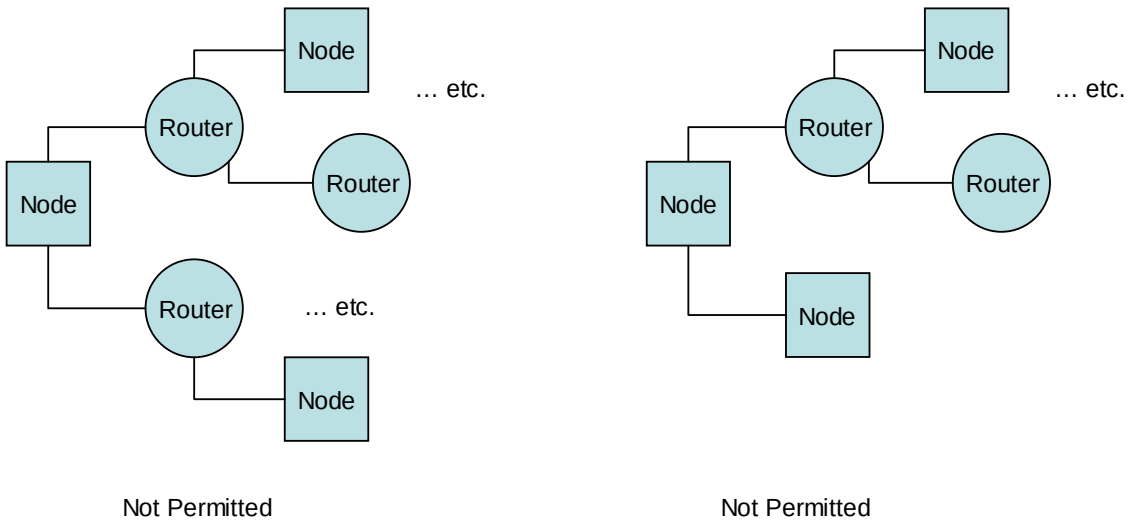


Figure 3-1: Multi-Port Router Topology Restrictions

These situations can be avoided by linking the two otherwise distinct networks, to form a single network. If a bridge is required, a unit must be created which appears to SpaceWire-PnP as two distinct nodes. The unit is then responsible for transferring data between the networks.

3.2.4 Network Size

In most cases SpaceWire-PnP does not place restrictions on the size of the SpaceWire network. However, on level 2 networks (see below) part of the network discovery algorithm needs to wait for a possible response from a device which may or may not still be present on the network. In the case where the device is no longer present, a reply from the device will never arrive. The algorithm must therefore use a timeout to protect against this case. To permit interoperability, an upper bound on this timeout must be specified in this protocol definition. This necessarily bounds the size of the network in terms of the maximum time it can take to obtain a reply from a device in response to a simple read command. This is determined by assuming the following:

- A maximum of 12 network hops (12 routers)
- A minimum link speed of 10Mbps
- A maximum router propagation delay of 1µs
- A maximum node response time (receipt of read command to transmission of read reply) of 100µs

This results in a maximum network size, in terms of command/reply time, of 260µs.

3.3 Protocol Composition and Structure

Devices can support SpaceWire-PnP to one of two levels (level 1 and level 2). The level of support determines how *open* the network can be. More information on this is presented in Section 3.3.2. For both levels of support, the structure of the various elements on SpaceWire-PnP is the same. This section introduces the various concepts behind the structure of the protocol.

3.3.1 Targets, Initiators and Active Nodes

The descriptions and definitions in this document will often refer to targets and initiators. In both cases these terms are used in the same way as in the RMAP standard [AD2]. An *initiator* is the source of an RMAP or SpaceWire-PnP command, such as a read or a write. A *target* is the destination for an RMAP or SpaceWire-PnP command, and is responsible for acting on the command and issuing a reply. For SpaceWire-PnP operations, an initiator must be a node, but a target can be any device (either a node or a router). Not all nodes will have the ability act as an initiator for

SpaceWire-PnP commands, those that can are referred to as *active nodes*, and those that cannot are known as *passive nodes*.

3.3.2 SpaceWire-PnP Support Levels

Every SpaceWire-PnP device supports SpaceWire-PnP to either *level 1* or *level 2*. The level of support of one or more active nodes on the network defines the level of the network as a whole. A network is always entirely level 1 or level 2; it is not possible for a portion of a network to be level 1 and another portion to be level 2.

Level 1 networks exist when there are no level 2 active nodes present, only one or more level 1 active nodes. Both level 1 devices and active nodes are simpler than those that support level 2. This simplicity is enabled by requiring that network designers take responsibility for ensuring that there is no competition between active nodes for configuration of any device one the network. In most cases this is carried out by ensuring that only one active node (or manager) is actually active at any time. Level 1 networks are therefore referred to as *managed networks*.

Level 2 devices are required to have a small number of additional facilities, and level 2 active nodes are required to have more sophisticated algorithms. These algorithms arbitrate between multiple active nodes, removing any strict requirement for *a priori* network design (although a designer can control exactly how the arbitration works). Level 2 networks can therefore cope with unexpected devices being added, including additional active nodes. Level 2 networks are therefore referred to as *open networks*.

As level 1 active nodes are intended to be managed by a designer, the management of these devices is not exposed as part of the SpaceWire-PnP protocol. For this reason level 1 and 2 active nodes cannot coexist on the same network. On the other hand, SpaceWire-PnP defines algorithms to permit other level 1 devices (passive nodes and routers) to participate on level 2 networks, albeit slightly less efficiently than level 2 devices, as these algorithms require additional timeout states to resolve competition situations.

3.3.3 Level 2 Networks: Owners and Proxies

Level 2 networks introduce a number of additional concepts not present in level 1 networks to permit the coexistence of multiple active nodes, and to allow a designer to exercise control over the way a network is managed when SpaceWire-PnP algorithms are resolving competition between multiple active nodes.

All devices on a fully discovered level 2 network have an *owner*. The owner is responsible for managing the device and providing all of the services offered by the device via a *proxy*. The proxy is a copy of all of the device parameters, hosted by the device owner and allows other active nodes an

interface for the configuration of the device. When an active node wishes to configure a device which it does not own, it directs the read or write command to the proxy, rather than the actual device. The owner then gets an opportunity to vet the command and either approve it (by carrying out the equivalent command on the actual device), modify it, perhaps without the original initiator knowing, or rejecting the operation outright.

Each device identifies its owner, a *proxy ID* and a *proxy key*. The ID and key are used to access the proxy of the device by specifying them as part of commands directed to the device owner, rather than device itself. This process resolves resource sharing issues caused by multiple accesses to the device. The ID specifies to the owner which proxy the operation refers to and allows an owner to proxy for up to 255 devices. The proxy key is intended to disambiguate between different devices and helps guard against the situation where a new device is added to a network and the proxy ID set in the device clashes with one that already exists. Except in cases where owners can use, for example, static device identifiers to ensure uniqueness, it is safest for owners to assign both the proxy ID and the proxy key randomly.

Device ownership must be contiguous across a network; this means that a node cannot claim ownership of a device unless it owns all routing devices on at least one path between it and the device. The device ownership algorithm also accounts for competition between prospective owners. This process can be controlled by a network designer via pre-assigned priorities. The SpaceWire-PnP competition mechanism normally results in regions being managed by a single device, except for in pathological cases, which can easily be prevented. In all situations, region ownership is fully deterministic.

3.3.4 Services, Parameters, Fields, Actions and Primitives

The features offered by SpaceWire-PnP are grouped together into *services*. There are two types of service provided by SpaceWire-PnP: *core* services, which relate to SpaceWire-PnP itself; and *capability* services, which are additional services providing control over additional capabilities offered by the device. Targets implementing a service normally provide the following:

- One or more *parameters* which may be read and/or configured. *Complex parameters* (such as tables) may have multiple *entries*, numbered contiguously from 0 to 255; *simple parameters* have only one entry, entry zero.
- Each parameter entry is, in turn, composed of one or more *fields*. Entry zero of a complex parameter is referred to as the *root entry*. The field meanings in each *non-root entry* of a complex parameter are always identical. The root entry provides different fields giving information applicable to all entries.

In some cases a target must obey certain algorithms when parameter fields are written or read. These algorithms are referred to as target *actions*.

Initiators expose access to target parameters via *primitives*, which also cover algorithms for processes such as network discovery.

Some services are mandatory, whilst others are optional. Even mandatory services may have optional parts, and parts which are only mandatory for level 2 devices. The mandatory/optional status is defined for each service, parameter, field, action and primitive in Sections 5 and 6.

3.4 USAGE OF THE REMOTE MEMORY ACCESS PROTOCOL

SpaceWire-PnP uses the Remote Memory Access Protocol (RMAP) standard [AD2] to define the packet format and the basic semantics of the protocol. To distinguish between generic RMAP packets and SpaceWire-PnP packets, a distinct protocol identifier is used for SpaceWire-PnP. SpaceWire-PnP also defines:

- The use of write command variants (verified, acknowledged write is always used).
- Incrementing address mode (incrementing mode is always used).
- For level 2 devices, the operation of read-modify-write instructions; level 1 devices need not support read-modify-write.
- The encoding of services, parameters, entries and fields into the RMAP address space, additionally, for level 2 active nodes only, the specification of a proxy identifier and proxy key.
- Addressing width (all fields are 32-bits wide).
- Endianness (byte-ordering is always big-endian).
- The key value to use when accessing both devices and, for level 2 active nodes only, device proxies.

A full description of the implementation of SpaceWire-PnP using RMAP is presented in Section 6.

3.5 LEVEL 1 SERVICES

There are five level 1 services:

1. Device Identification
2. Network Management
3. Link Configuration
4. Router Configuration

5. Time-Code Source

The Router Configuration service only applies to routers and the Time-Code Source service is optional (for devices that can act as a source for time-codes).

3.5.1 Device Identification

The Device Identification service provides basic information about the identity, status and capabilities of a device. All parameters associated with this service are read-only.

The Device Information parameter contains fields specifying:

- The Vendor ID for this device. Vendor IDs are assigned by the SpaceWire Working Group and are guaranteed to be unique for each vendor.
- The Product ID for this product type. This is specified by the vendor.
- Whether this device is a router or a node.
- SpaceWire-PnP level support (1 or 2).
- The number of ports this device has.
- An optional preferred network region for this device which, if provided, can aid a network manager with logical address assignment.
- An optional static Device ID, which can be up to 64-bits. If provided, this ID must be network unique.
- The instance of this device type in the system. This is to allow network managers to distinguish between otherwise identical devices, such as prime and redundant units.
- The length of read and write commands supported by this device.

The Device Identification service also permits a device to specify character strings describing the vendor and device type. These strings are encoded in UTF-8 and are optional. It is expected that such strings would be most useful for laboratory equipment and EGSE.

The Device Status parameter gives the current status of the device. This shows which ports are active and the identifier that has been assigned to this device to permit network discovery. An operational status field is also present which allows a device to indicate whether it is functioning correctly. The Device Status parameter also indicates the number of the port through which the device is being accessed. This information is provided to permit network discovery.

The Capability List parameter provides a simple list of protocols which this device supports. Where a protocol is able to encapsulate or transport another protocol, or the definition of the protocol permits

further standardisation (as is the case with RMAP, where standardisation of address spaces is possible), the capability list permits the specification of a "transport protocol". The use of this facility can be explained with two examples:

- Suppose a device supports SpaceWire-RT and a Packet Transfer Protocol (PTP), but that the PTP is only supported over SpaceWire-RT. In this case the Capability List specifies PTP as a supported protocol, with SpaceWire-RT as the transport protocol.
- Suppose a device supports RMAP with standard format data sources and sinks. In this case RMAP is specified as the transport protocol and the "supported protocol" part refers to the fact that standard data sources and sinks are supported.

Each capability entry may have an associated Capability Service to permit configuration of the supported protocol. Capability Services are covered in more detail in Section 3.7

Initiator primitives for the Device Identification service are very simple and support the reading of all parameters.

3.5.2 Network Management

The Network Management service permits access to parameters necessary for discovering and managing a network. For level 1 networks the Network Management service is very simple, relying on the fact that there is guaranteed to be only one active node (or manager) at any given time, or that competition between active nodes is resolved by design.

For level 1 devices only one parameter is used for the Network Management service: Network Identification. This parameter provides a read-write field to allow devices to be uniquely identified during network discovery. Unique identification is necessary to determine the existence of network paths which form loops.

A second field is optionally provided to allow the device to be assigned a logical address. This permits the device to be able to source commands, such as RMAP commands, and specify a valid source address. If this field is not provided it is read-only and zero.

Initiator primitives associated with the Network Management service permit the Network Identification parameter to be read and written, as well as a breadth-first traversal of the network, identifying all devices. SpaceWire-PnP uses a breadth-first traversal exclusively as research has shown that it gives faster results in most networks, consumes the least network bandwidth, and provides better tolerance of network changes [RD3]. It is also consistent with level 2 networks which must use a breadth-first traversal to permit active node conflict resolution (see Section 3.6 for more information). Initiator primitives are also provided to allow the logical address field to be configured.

3.5.3 Link Configuration

The Link Configuration service permits the various features of all links on a device to be queried and configured. SpaceWire-PnP supports two types of link: SpaceWire links, which have a transmit speed and can be started and stopped as stated by the SpaceWire standard [AD1]; “external ports” which are viewed as being always-connected, fixed speed ports. External ports are typically used to connect on-chip or on-board devices together, often via a parallel rather than serial, connection.

The setting of link transmit speed is flexible to permit a range of possible implementation techniques, and to provide support for the transmit rate generation mechanisms used in current hardware such as the SpW-10X. The transmit rate of a given link is determined by the value of two independent fields:

- The Reference Rate, which is a rate common to all links.
- The Link Rate, which is based on the Reference Rate, is a different field for each link and determines the transmit rate for that link.

Each rate field is accompanied by a field specifying the maximum and minimum permissible values for the rate setting. Devices state whether the Reference Rate and Link Rate should be specified as a value in Mbps or as a divider. Where the Reference Rate is specified as a divider, a read-only value gives the Base Rate, which is the transmit rate (in Mbps) before division by the Reference Rate divider. Where both the Reference Rate and Link Rates are specified as dividers, the transmit rate for a single link is determined as follows:

- The Base Rate (a constant value) is divided by the Reference Rate divider value to give the current reference rate.
- The current reference rate is divided by the Link Rate divider value to give the transmit rate for that link.

In the case where the Reference Rate is specified as a value in Mbps, the Base Rate value is unused (it simply specifies the maximum permissible value for the Reference Rate). Where the Link Rate is specified as a value in Mbps, the maximum and minimum permissible values for the link are updated to reflect changes in the Reference Rate.

Although both Reference Rate and Link Rate fields are provided, a device does not have to support these functions. A device may support neither (a fixed rate link), Reference Rate only (all links have the same speed), Link Rate only (all link speeds are independent), or both.

The Link Configuration service provides the following parameters:

- Port Activity, which provides fields to determine the current status of each port (connected or disconnected) and the change in status (the delta) since the status was last read.

- Reference Rate, which allows the configuration of the reference transmit rate (see explanation above).
- Link Control, which is a complex parameter with an entry for each link on the device.

Each Link Control parameter entry has fields to determine:

- The type of each link (SpaceWire or external).
- The status of the link (whether there have been any errors on the link, and whether it is connected or not).
- The maximum and minimum permissible Link Rate values.
- The current Link Rate value.
- The priority of this port, for arbitration (this only applies to routers and is described in more detail in Section 3.5.4).
- The state of the link (allowing starting and stopping of the link, and the autostart mode specified in the SpaceWire standard [AD1]).
- Control over whether or not time-codes should be transmitted/propagated on this link.

Initiator primitives associated with the Link Configuration service are very simple, permitting queries of the read-only parameters and configuration of the read-write parameters.

3.5.4 Router Configuration

Target parameters associated with the Router Configuration service are only provided by routers and permit the various features of a router to be queried and configured. This is enabled by two parameters: Router Control, which controls the routing mechanisms used by this device; and the Routing Table which permits configuration of address-specific settings for both path (physical) and logical addresses.

The routing mechanism used by the router is configured by two fields of the Router Control parameter: Watchdog Timer and Arbitration Mechanism, optionally combined with the priority fields associated with each link (see Section 3.5.3, the Link Control parameter) and with each address (see discussion of the Routing Table parameter, below).

The Watchdog Timer is a value in microseconds which is used to protect the router, and network, against blocked links. Each link has an associated timer which is reset every time data is transmitted on that link. If the watchdog time is reached after data has been transmitted, but before an EOP has been sent, it is assumed that the link is blocked. In this case, the remainder of the packet arriving at the router is spilt (consumed and discarded), and an EEP is attached to the portion of the packet

remaining on the blocked link. The operation of the watchdog timer is identical to that used by the SpW-10X. It is not required for all routers to implement a watchdog timer; if the timer is not implemented, the Watchdog Timer field is read-only and set to a value which indicates infinity (a timeout will never be reached).

The Arbitration Mechanism field permits a high level of flexibility in how the router arbitrates between incoming packets competing for the same output port. There are three mechanisms supported:

- Link arbitration, where the configurable priority value associated with each port is used to determine which packet should go first (highest priority wins).
- Address arbitration, where the configurable priority value associated with each address in the routing table is used to determine which packet should go first (again, highest priority wins).
- Round robin arbitration, where each port waiting to transmit a packet to an output port is given equal priority and each port with an incoming packet is given a chance to transmit to the output port in turn.

The three arbitration mechanisms can each be enabled and disabled separately, and can be combined. Where more than one arbitration mechanism is enabled, the mechanisms are applied in the order specified above, with a lower arbitration mechanism only used if a higher one fails to resolve the conflict. If the enabled arbitration mechanisms fail to resolve the conflict, then a fixed arbitration scheme is used, where the physical port number of the packet waiting acts as a priority, where lower port numbers have higher priority than higher ones. As with the watchdog timer, there is no requirement for routers to support any of the non-fixed arbitration mechanisms. A router may implement one, two, or all three of the mechanisms, as desired.

The Time-Code Counters field of the Router Control parameter enables the current value of up to four independent time-code counters to be enabled, read and reset. Each time-code counter corresponds to a time-code channel, or a setting of the two most significant bits in a time-code value. As the SpaceWire standard only requires support for channel zero, all other channels in this field are optional. A system may choose not to support other channels, or may be using other channels for a different purpose.

The Routing Table complex parameter has an entry for each path (physical) address, and each supported logical address. A device can choose not to support the full range of logical addresses but, as with ports, the supported range must be contiguous starting from the lowest permissible address, which is 32 for logical addresses. The root entry specifies the number of entries in the routing table. There will always be an entry for each path address, and entries corresponding to path addresses which do not relate to ports on the device are reserved. Entries from 32 upwards are for logical

addresses. Each non-root entry in the routing table has two fields which control how packets with the corresponding address are routed:

- The Port Association field specifies which output ports are to be used for routing packets with this address. Each bit corresponds to an output port with that number; for example, bit 1 corresponds to port 1. When the entry is for a physical port, the bit specifying the corresponding output port (so bit 1 for the entry for address 1) is always set, and is read-only. This is to prevent confusing situations arising where a path addressed packet may not be routed as expected. A device may permit more than one bit to be set, in which case either Group Adaptive Routing (GAR) or Packet Distribution (PD) is used, according to the settings in the Address Control field.
- The Address Control field specifies the action used when routing packets with the corresponding address. The address may be enabled and disabled; and the priority for arbitration may be set (if supported). The routing action bit specifies whether GAR or PD is to be used when more than one bit is set in the Port Association field. A bit in the Address Control field is optionally used to enable the spilling of packets if no output port is available (or if one of the specified output ports is not available, in the case of PD). This is useful if, by design, links should never be blocked, and a blockage corresponds to a failure case. Spilling a blocked packet would then remove it from the network immediately.

When using GAR, an incoming packet is sent out of the first available output port specified in the Port Association field. If PD is specified, the router blocks until all specified output ports are available, and then duplicates the packet onto all of the specified ports. Support for both GAR and PD is optional: a router may not permit more than one bit to be set in the Port Association field; if it does, it may choose to only support either GAR or PD by ensuring that the routing action bit is read-only.

As with the Link Configuration service, initiator primitives associated with the Router Configuration service are very simple, permitting queries of the read-only parameters and configuration of the read-write parameters.

3.5.5 Time-Code Source

The Time-Code Source service is entirely optional and allows devices that have the ability to act as a time-code master to be configured across the network. The service provides one complex parameter, called Time-Code Sources, which has one entry for each source supported by this device. A device may support up to four time-code sources, each of which can be configured independently.

A time-code source can generate time-codes in two ways:

- Single-shot, where a single time-code is sent in response to a write operation on the Generation Control field of the Time-Code Sources parameter. The value of the time-code can either be a value specified in the write, or the next consecutive value based on the time-code counter.
- Periodic, where a periodic time is enabled which sends time-codes at regular intervals. Time-codes will always have consecutive values.

The time-code counters associated with each source can also be reset, and the channel on which time-codes are transmitted (the value of the two most significant bits in the time-code) can be controlled. Some time-code sources may choose to limit the channels to which they can be assigned by making these bits read-only.

Initiator primitives associated with the Time-Code Source service are very simple, configuration of all parameters.

3.6 LEVEL 2 SERVICES

Support for SpaceWire-PnP level 2 does not require any further services; however, there are three differences between level 1 and level 2 service support:

- Support for the read-modify-write RMAP command is required.
- An Owner Data parameter is added to the Network Management service, which identifies the owner of the device and proxy information. Owners may also provide information about the local network topology using the Network Management service.
- Owners are required to act as proxies for the Router and Link Configuration services, the Time-Code source service and all capability services. Active nodes are required to use the proxy, rather than the device itself, for these services.

Differences between level 1 and level 2 services are described in the following sections.

3.6.1 Device Identification

The Device Identification service for Level 2 devices provides one extra field as part of the Device Status parameter. This field provides a mirror of the Device Owner field provided as part of the Network Identification parameter of the Network Management service. The level 2 initiator primitive associated with the Device Status parameter provides access to this additional field.

As all Device Identification parameters are read only, there is no requirement to access these parameters using the owner proxy, even on a level 2 network.

3.6.2 Network Management

The level 2 Network Management service provides facilities to permit the discovery and management of open networks. Device targets must support an additional field, Device Owner, as part of the Network Identification field. For routing devices only, this field is complemented by an action which configures a routing table entry for the device owner when this field is written.

The Device Owner field identifies the location of the owner, either by logical or path addressing, and the proxy ID and key used for accessing the owner proxy for this device.

Device owners must provide one additional field when the Network Identification parameter is accessed via the owner proxy. This field, which specifies the region and priority of the owner of the device, is used for ownership competition resolution.

Owner proxies may optionally provide the Local Topology complex parameter, which presents the topology of the local network in a form similar to that which is provided by the network discovery primitives. This parameter permits active nodes to share topology information, reducing network traffic during discovery operations involving multiple active nodes. Each non-root entry in the Local Topology parameter represents a device on the network and lists, for each physical port on the device, the other device to which it is connected. The connections are specified by referencing the entry number representing the device to which this device is connected to and the port on the remote device which is used for the connection.

Initiator primitives are provided to support all network management operations:

- Owner identification and verification of network presence;
- Device claim, to attempt to claim ownership of devices;
- Competition resolution, to resolve competition between active nodes when attempting to claim ownership of a device;
- Claim release, to release ownership of a device;
- Generate proxy, which creates a proxy for a specified device;
- Proxy release, which releases all resources associated with a device proxy;
- Network discovery, which discovers devices on a level 2 network.

Owner proxies act as targets for SpaceWire-PnP commands. The operation of proxy targets is defined by the Proxy Provision action and is complemented by the Proxy Redirection action which defines how initiators should access and address proxies. These actions define how proxy IDs and proxy keys are used, and how return addresses should be formed.

3.6.3 Link Configuration

To prevent resource-sharing issues, all Link Configuration service parameters must be accessed via the owner proxy, unless the initiator is the device owner.

3.6.4 Router Configuration

To prevent resource-sharing issues, all Router Configuration service parameters must be accessed via the owner proxy, unless the initiator is the device owner.

3.6.5 Time-Code Source

To prevent resource-sharing issues, all Time-Code Source service parameters must be accessed via the owner proxy, unless the initiator is the device owner.

3.7 CAPABILITY SERVICES

Capability services are SpaceWire-PnP services to permit the interrogation and configuration of capabilities. A capability is a SpaceWire protocol or protocol/transport combination, as described in Section 3.5.1. The SpaceWire-PnP protocol definition defines two standard capability services which specify RMAP as the transport protocol: data source and data sink.

3.7.1 RMAP Data Sources

The Data Source capability service permits a device to expose data of any type using RMAP in a standard way. The Data Source capability service (using SpaceWire-PnP) is used to detect and configure available data sources. The actual data is obtained using RMAP, not SpaceWire-PnP. In this manner, the Data Source service can be used to describe data within a pre-existing RMAP address space. Similarly, existing IP for accessing data over RMAP can be used in a SpaceWire-PnP system. The Data Source capability service permits the description and configuration of both RMAP targets and RMAP initiators, with support for up to 255 of each.

Each source (either initiator or target) can only be used by one device at any given time. To protect against the case where a device begins using a data source, and the either leaves the network or ceases to function, each source has a *lease timer* which indicates whether the source is in use, and which expires after a given time, thus freeing up the data source. The lease time begins when the lease timer is set, and is reset whenever a source target is read or an initiator target receives a reply. It is not necessary to use the lease timer.

A data source target may support a queue of pending read operations. If a queue is supported, a reply to an RMAP read command addressed to the data source may not be issued until there is data

available. A reply timeout may also be specified which acts as a watchdog: if a reply has not been issued by the time the reply timeout elapses a reply will be generated anyway. This can be useful to inform an active node that a data source target is still present on the network and functioning correctly.

Using these features, a simple target data source, with no queue, can be used to describe a pre-existing RMAP address space, such as an instrument. The queue feature permits delayed-response reads to be used to notify an active node when new data is available. One such use for this feature is for routers to be able to inform active nodes when the status of the active links has changed i.e. when a link has become available or unavailable. This would correspond to a hot-plug event used in other plug-and-play technologies.

Initiator data sources follow a similar pattern: SpaceWire-PnP parameters may be used to configure RMAP write commands initiated by the data source. Such commands may be acknowledged or unacknowledged. A reply timeout on an initiator data source can be used to configure the data source to retry RMAP writes if either they are not acknowledged (for acknowledged writes), or if the reply timeout is not re-written. In order to facilitate the matching of incoming RMAP replies to data sources which are waiting for replies, the bottom 8 bits of the transaction ID of initiated writes is defined by the data source, and would usually contain the initiator data source index; the top 8 bits are configurable.

3.7.2 RMAP Data Sinks

RMAP data sinks function in exactly the same manner as RMAP data sources except in the capacity that they are responsible for sinking data rather than sourcing it. In this way target data sinks accept RMAP write commands and initiator data sinks initiate RMAP read commands.

4 SpaceWire-PnP Networks

This section describes the requirements that SpaceWire-PnP places on devices and networks. This section is normative: if these requirements are not met, the resulting network may not function correctly.

4.1 Port Provision and Numbering

4.1.1 Routers

Routers shall provide at least two ports, not including the configuration port. The maximum number of ports a router shall provide is 31 [AD1].

4.1.2 Nodes

Nodes shall provide at least one port. The maximum number of ports a node shall provide is 31 [AD1].

4.1.3 Port Numbering

On all devices, ports shall be numbered contiguously from 1 up to the number of ports supported by the device.

4.2 Routing Function

4.2.1 Routers

Routers shall provide a routing function in accordance with the SpaceWire standard. This shall include support for path addressing and may include support for logical addressing.

Packets arriving at a router with a path address of zero shall be interpreted as containing configuration information.

4.2.2 Nodes

Nodes shall not provide a routing function. Packets arriving at a node with a path address of zero shall be interpreted as containing configuration information. The behaviour of a node receiving packets with any other addressing information shall be vendor specific; however, the node shall not use addressing information to route packets between ports.

4.3 Network Reachability

If the network must be fully discoverable, any connected ports on any node shall be reachable from any other port on any node on the network, via routers, if appropriate.

4.4 Level 1 and Level 2 Device Coexistence

A level 1 network is defined as a SpaceWire-PnP network with at least one level 1 active node and no level 2 active nodes. A level 2 network is defined as a SpaceWire-PnP network with at least one level 2 active node. Level 1 active nodes shall not be present on level 2 networks. Level 1 routers and passive nodes shall be permitted on both level 1 and level 2 networks; likewise, level 2 routers and passive nodes shall be permitted on both level 1 and level 2 networks.

4.5 Network Size and Timing

For level 2 networks only, the size of the network shall be limited by the maximum network propagation delay between transmitting the EOP of a SpaceWire-PnP read command packet requesting a 4 byte (1 word) read and receiving the EOP of the associated read reply packet. This propagation delay shall be limited to 260µs.

5 LEVEL 1 SERVICE DEFINITIONS

This section defines the various services offered by level 1 SpaceWire-PnP devices, detailing their parameters, target actions and primitives. The mandatory/optional status of each service, parameter and primitive is presented.

5.1 SERVICES OVERVIEW

SpaceWire-PnP provides up to 5 services at level 1, these are outlined in Table 5-1.

Table 5-1: Level 1 SpaceWire-PnP Services

ID	Name	Summary
0	*Reserved*	*Reserved for future use*
1	Device Identification	Provides core information to allow the identification of the device, and its operational status
2	Network Management	Provides mechanisms for discovering and managing network resources.
3	Link Configuration	Provides the ability to query the status of any device link and configure its state and speed
4	Router Configuration	This service is applicable to routing devices only. Provides support for the configuration of routing tables and routing mechanisms, as well as time-code propagation
5	Time-Code Source	This service, if implemented, provides a standard method for exposing a time-code source, allowing time-code generation to be enabled/disabled, the frequency of time-codes to be configured, and ticks to be generated manually, if appropriate
11-31	*Reserved*	*Reserved for future use*

The following sections detail each of the services in turn.

5.2 DEVICE IDENTIFICATION SERVICE

5.2.1 Service Provision

Provision of the Device Identification service is mandatory for both targets and initiators.

5.2.2 Target Parameters

Device Identification parameters are summarised in Table 5-2.

Table 5-2: Device Identification Service Parameters			
ID	**Name**	**Type**	**Summary**
0	Device Information	Simple	Provides essential device information
1	Vendor String	Simple	Optional UTF-8 vendor description string
2	Product String	Simple	Optional UTF-8 product description string
3	Device Status	Simple	Provides current device status
4	Capability List	Complex	Provides a list of supported protocols
5-7	*Reserved*	-	*Reserved for future use*

5.2.2.1 Device Information

Device Information is a simple parameter comprised of the fields summarised in Table 5-3. Fields in the Device Information parameter shall be read-only and constant.

Table 5-3: Device Information Parameter Fields		
ID	**Name**	**Summary**
0	Vendor ID/ Product ID	Contains 16-bit vendor and product IDs
1	Region/Number of Ports	Indicates preferred device region gives port count
2	Static Device ID High	High 32 bits of the 64-bit static device ID (if present)
3	Static Device ID Low	Low 32 bits of the 64-bit static device ID (if present)
4	Version/Instance ID	Version and System instance of this device type
5	Operation/String Lengths	Length of the vendor a product strings (can be zero)
6-31	*Reserved*	*Reserved for future use*

5.2.2.1.1 Vendor ID/Product ID Field

The Vendor ID/Product ID field shall comprise a 16-bit vendor ID and a 16-bit product ID. The value of the vendor ID shall be used in accordance with vendor IDs allocated and assigned by the SpaceWire Working Group. A vendor shall use their assigned ID and shall not use any other ID. The

value of the product ID shall be chosen by the vendor. The Vendor ID/Product ID field is encoded as shown in Figure 5-1.

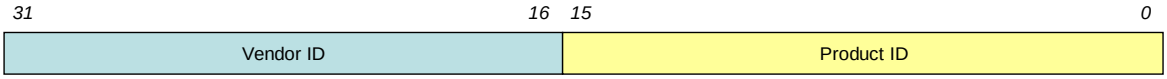


Figure 5-1: Vendor ID/Product ID Field Encoding

5.2.2.1.2 Region/Number of Ports Field

The Region/Number of Ports field (see Figure 5-2) shall have three parts:

- Bits 8 to 31 (where 8 is the least significant) shall specify the preferred region for this device. If the device does not wish to specify a preferred region, this shall be zero.
- Bit 7 shall identify whether the device is a node or a router. Node devices shall specify 0 for this field; routing devices shall specify 1.
- Bit 5 shall identify the protocol support level offered by this device. If this device supports SpaceWire-PnP level 2, this bit shall be set to 1; 0 shall indicate that device support is limited to SpaceWire-PnP level 1.
- Bits 0-5 (where 0 is the least significant) shall specify the number of ports the device has, not including the configuration port.

Bit 6 is reserved for future use; the value of this bit shall be ignored.

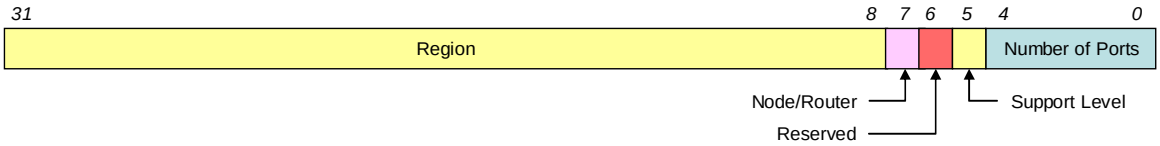


Figure 5-2: Number of Ports/Region Field Encoding

5.2.2.1.3 Static Device ID Fields (Low and High)

The Static Device ID Low field shall contain the lowest 32-bits of a 64-bit globally unique device identifier (with bit 0 of the device ID assigned to bit 0 of the field) or zero, if a device identifier is not specified. The Static Device ID High field shall contain the highest 32-bits of a 64-bit globally unique device identifier (with bit 32 of the device ID assigned to bit 0 of the field) or zero, if a device identifier is not specified.

If the device specifies a device identifier it shall be guaranteed to be network unique. If the device ID specified is 32-bit, or shorter, the identifier shall be placed in the Static Device ID Low field and the

Static Device ID High field shall be zero. If a device identifier is not provided, both Static Device ID fields (Low and High) shall be zero.

5.2.2.1.4 Version/Instance ID Field

The Version/Instance ID field (see Figure 5-3) shall have four parts:

- Bits 24 to 31 (where 24 is the least significant) shall hold the major version. This shall be an unsigned number from 0 to 255, assigned by the vendor.
- Bits 16 to 23 (where 16 is the least significant) shall hold the minor version. This shall be an unsigned number from 0 to 255, assigned by the vendor.
- Bits 8 to 15 (where 8 is the least significant) shall hold the patch number. This shall be an unsigned number from 0 to 255, assigned by the vendor.
- Bit3 0 to 7 (where 0 is the least significant) shall hold the instance ID. The instance ID permits otherwise identical network devices to be distinguished (for example, left and right star trackers). The instance ID shall be configurable by the product integrator in a vendor defined manner.

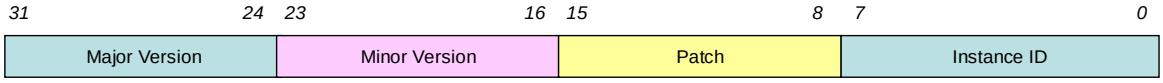


Figure 5-3: Version/Instance ID Field Encoding

5.2.2.1.5 Operation/String Lengths Field

The Operation/String Lengths field specifies the maximum supported lengths of read and write operations, as well as the lengths (in bytes) of the vendor and product strings (supplied in the Vendor String and Product String parameters respectively). The encoding of the Operation/String Lengths field shall be as shown in Figure 5-4.

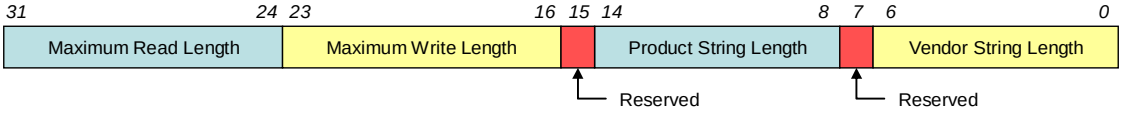


Figure 5-4: Operation/String Lengths Field Encoding

The Operation/String Lengths shall have four parts:

- Bits 24 to 31 (where 24 is the least significant) shall contain the maximum read operation length supported by this device in multiples of 32-bit words.

- Bits 16 to 23 (where 16 is the least significant) shall contain the maximum write operation length supported by this device in multiples of 32-bit words.
- Bits 8 to 14 (where 8 is the least significant) shall contain the length of the product string provided by the Product String parameter, in bytes.
- Bits 0 to 6 (where 0 is the least significant) shall contain the length of the vendor string provided by the Vendor String parameter, in bytes.

Bits 7 and 15 are reserved for future use; the value of these bits shall be ignored.

5.2.2.2 Vendor String

Vendor String is a simple parameter comprised of 0 to 32 fields containing a UTF-8 encoded string (see [AD4]) providing a description of the vendor of this device. The length of the string, in bytes, is described by the Operation/String Length field of the Device Information parameter. If the vendor string length is greater than 0, the first byte of the string shall be accessible in bits 24 to 31 of field ID 0; the second byte shall occupy bits 16 to 23; the third byte shall occupy bits 8 to 15 and the fourth shall occupy bits 0 to 7. Each byte shall be encoded such that the lowest numbered bit is the least significant. The fifth byte shall occupy bits 24 to 31 of field ID 1, and so on. The vendor string shall occupy a whole number of 32-bit fields. Extra trailing bytes shall be padded with zeros.

5.2.2.3 Product String

Product String is a simple parameter comprised of 0 to 32 fields containing a UTF-8 encoded string (see [AD4]) providing a description of the product type of this device. The length of the string, in bytes, is described by the Operation/String Length field of the Device Information parameter. If the product string length is greater than 0, the first byte of the string shall be accessible in bits 24 to 31 of field ID 0; the second byte shall occupy bits 16 to 23; the third byte shall occupy bits 8 to 15 and the fourth shall occupy bits 0 to 7. Each byte shall be encoded such that the lowest numbered bit is the least significant. The fifth byte shall occupy bits 24 to 31 of field ID 1, and so on. The product string shall occupy a whole number of 32-bit fields. Extra trailing bytes shall be padded with zeros.

5.2.2.4 Device Status

Device Status is a simple parameter comprised of the fields summarised in Table 5-4. Fields in the Device Status parameter shall be read-only.

Table 5-4: Device Status Parameter Fields		
ID	**Name**	**Summary**
0	Network ID	Mirror of Network ID field
1	Active Ports	Mirror of Active Ports field
2	Return Port/Operational Status	Current return port number and device status
3-31	*Reserved*	*Reserved for future use*

5.2.2.4.1 Network ID Field

The Network ID field of the Device Status parameter shall contain a read-only mirror of the contents of the Network ID field of the Network Discovery parameter provided by the Network Management service (see Section 5.3.2.1.1).

5.2.2.4.2 Active Ports Field

The Active Ports field of the Device Status parameter shall contain a mirror of the contents of the Active Ports field of the Port Activity parameter provided by the Link Configuration service (see Section 5.4.2.1.1).

5.2.2.4.3 Return Port/Operational Status Field

The Return Port/Operational Status field (see Figure 5-5) shall contain two parts:

- Bits 8 to 12 (where 8 is the least significant) shall identify the number of the port which will be used to reply to this request.
- Bits 0 to 7 (where 0 is the least significant) shall provide an unsigned number from 0 to 255 identifying the operational status of the device. Zero shall indicate that the device is functioning correctly. A non-zero value shall indicate that the device is not functioning correctly. The meaning of non-zero values shall be vendor-specific.

Bits 13 to 31 are reserved for future use; the value of these bits shall be ignored.

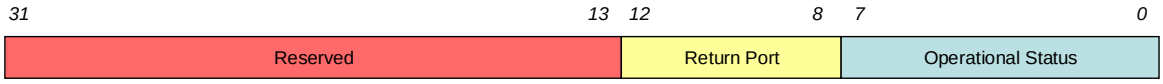


Figure 5-5: Return Port/Operational Status Field Encoding

5.2.2.5 Capability List

Capability List is a complex parameter providing a list of capabilities supported by this device. All fields in the Capability List parameter shall be read-only and constant. Fields in the root entry are summarised in Table 5-5.

Table 5-5: Capability List Parameter Root Entry Fields

ID	Name	Summary
0	Capability Count	Count of specified capabilities
1-31	*Reserved*	*Reserved for future use*

Capability List parameter may have up to 7 non-root entries, depending on the number of capabilities supported by this device. Each field in a non-root entry shall contain a Capability Record describing a capability offered by this device. There shall be as many Capability Records as specified by the Capability Count field. The first 32 capabilities shall be described by entry ID 1; the next 32 capabilities shall be described by entry ID 2 and so on. Within each entry, the first Capability Record shall occupy field ID 0; the second shall occupy field ID 1 and so on. Valid Capability Record fields shall be arranged contiguously, starting from entry ID 1, field ID 0.

5.2.2.5.1 Capability Count Field

The Capability Count field shall consist of an 8-bit number specifying the number of valid capability records present. The count shall be represented as an unsigned number from 0 to 255, with the least significant bit being bit 0. The field encoding is shown in Figure 5-6. Bits 8 to 31 are reserved. The value of these bits shall be ignored and shall always be written as zero. These bits are reserved for future use.

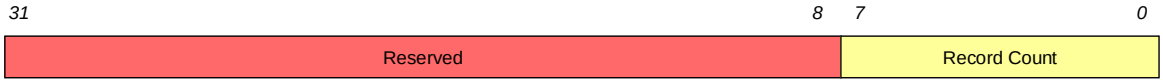


Figure 5-6: Capability Count Field Encoding

5.2.2.5.2 Capability Record Fields

Each Capability Record field shall specify a protocol (by protocol ID, or other suitable identifier) that this device supports. The record may also specify that the protocol is only valid when transported by another protocol. The meaning of “transported” here depends on the protocols concerned. Capability Record fields shall be encoded as shown in Figure 5-7.

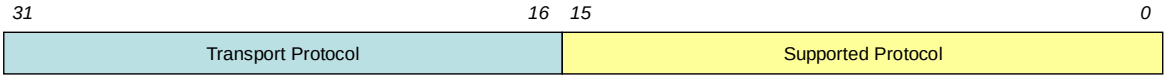


Figure 5-7: Capability Record Field Encoding

The Supported Protocol (bits 0 to 15, where 0 is the least significant) shall contain the protocol ID of a protocol that this device supports. If a transport protocol is required for the support of the Supported Protocol, this shall be specified by Transport Protocol (bits 0 to 15, where 0 is the least significant), which shall specify the protocol ID of the transport protocol. If no transport protocol is required, Transport Protocol shall be zero. A transport protocol shall also be included as a Supported Protocol. Where a Transport Protocol is specified, the Supported Protocol may not be a valid protocol ID, and may be used to describe a further level of standardisation of the Transport Protocol. This is used, for example, to describe standard RMAP address spaces.

Capability Record fields shall be ordered numerically, i.e. supported protocols with no transport protocol (Transport Protocol set to zero) shall be listed first (from entry ID 1, field ID 0 onwards).

SpaceWire-PnP may not be listed as a supported protocol; this does not indicate that SpaceWire-PnP is not supported. This is to permit devices that do not wish to publish any capabilities to have no Capability Records.

5.2.3 Target Parameter Provision

Device targets shall provide parameters in accordance with the rules shown in Table 5-6. “M” indicates that provision of the parameter/field for the identified operation is mandatory; “O” indicates that provision of the parameter/field for the identified operation is optional; “F” indicates that provision of the parameter/field for the identified operation is forbidden.

Table 5-6: Device Identification Service Target Parameter Provision

Parameter	Field(s)	Read	Write	RMW
Device Information	*All*	M*	F	F
Vendor String	*All*	O*	F	F
Product String	*All*	O*	F	F
Device Status	*All*	M	F	F
Capability List	*Root entry fields*	M*	F	F
Capability List	*Non-root entry fields*	O*	F	F

* All field values shall be constant.

5.2.4 Target Actions

A target shall not carry out any further actions beyond support for read/write/read-modify-write commands specified in Section 5.2.3

5.2.5 Initiator Primitives

A summary of Device Identification service initiator primitives is presented below:

- DIDS_READ_INFO.request
- DIDS_READ_INFO.indication
- DIDS_READ_VENDOR_STRING.request
- DIDS_READ_VENDOR_STRING.indication
- DIDS_READ_PRODUCT_STRING.request
- DIDS_READ_PRODUCT_STRING.indication
- DIDS_READ_STATUS.request
- DIDS_READ_STATUS.indication
- DIDS_READ_CAPABILITY_LIST.request
- DIDS_READ_CAPABILITY_LIST.indication

These primitives are considered in more detail in the following sections.

5.2.5.1 DIDS_READ_INFO.request

The DIDS_READ_INFO.request primitive shall permit an initiator to request device information for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

5.2.5.2 DIDS_READ_INFO.indication

The DIDS_READ_INFO.indication primitive shall provide an indication to an initiator that device information may be available in response to a DIDS_READ_INFO.request. This primitive shall indicate whether the request was successful or not, and may contain device information.

The parameters for this primitive shall be:

- Result
- Vendor_ID
- Product_ID
- Preferred_Region
- Router_Node
- Support_Level
- Port_Count
- Device_ID
- Version
- Instance_ID

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then all other parameters may be invalid.

Vendor_ID shall contain the vendor ID of the device (see Section 5.2.2.1.1).

Product_ID shall contain the product ID of the device (see Section 5.2.2.1.1).

Preferred_Region shall contain the preferred region identifier for the device; this may be zero if no region is preferred (see Section 5.2.2.1.2).

Router_Node shall indicate whether the device is a router or a node (see Section 5.2.2.1.2).

Support_Level shall indicate whether this device supports SpaceWire-PnP level 1 or 2 (see Section 5.2.2.1.2).

Port_Count shall contain the number of physical ports on the device (see Section 5.2.2.1.2).

Device_ID shall contain the device ID; this may be zero if no device ID is provided (see Section 5.2.2.1.3).

Version shall contain the version information for the device (see Section 5.2.2.1.4).

Instance_ID shall contain the instance ID for the device (see Section 5.2.2.1.4).

5.2.5.3 DIDS_READ_VENDOR_STRING.request

The DIDS_READ_VENDOR_STRING.request primitive shall permit an initiator to request the UTF-8 encoded vendor string for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

5.2.5.4 DIDS_READ_VENDOR_STRING.indication

The DIDS_READ_VENDOR_STRING.indication primitive shall provide an indication to an initiator that a UTF-8 encoded vendor string may be available in response to a DIDS_READ_VENDOR_STRING.request. This primitive shall indicate whether the request was successful or not, and may contain a vendor string.

The parameters for this primitive shall be:

- Result
- Vendor_String

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Vendor_String parameter may be invalid.

Vendor_String shall contain the UTF-8 encoded vendor string for the device; this may be an empty string if the device does not provide a vendor string (see Section 5.2.2.2).

5.2.5.5 DIDS_READ_PRODUCT_STRING.request

The DIDS_READ_PRODUCT_STRING.request primitive shall permit an initiator to request the UTF-8 encoded product string for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

5.2.5.6 DIDS_READ_PRODUCT_STRING.indication

The DIDS_READ_PRODUCT_STRING.indication primitive shall provide an indication to an initiator that a UTF-8 encoded product string may be available in response to a DIDS_READ_PRODUCT_STRING.request. This primitive shall indicate whether the request was successful or not, and may contain a product string.

The parameters for this primitive shall be:

- Result

- Product_String

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Product_String parameter may be invalid.

Product_String shall contain the UTF-8 encoded product string for the device; this may be an empty string if the device does not provide a product string (see Section 5.2.2.3).

5.2.5.7 DIDS_READ_STATUS.request

The DIDS_READ_STATUS.request primitive shall permit an initiator to request the device status for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

5.2.5.8 DIDS_READ_STATUS.indication

The DIDS_READ_STATUS.indication primitive shall provide an indication to an initiator that device status information may be available in response to a DIDS_READ_STATUS.request. This primitive shall indicate whether the request was successful or not, and may contain device status information.

The parameters for this primitive shall be:

- Result
- Network_ID
- Active_Ports
- Return_Port
- Operational_Status

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then all other parameters may be invalid.

Network_ID shall contain the current network ID of the device (see Section 5.2.2.4.1).

Active_Ports shall contain a list of currently active ports on the device (see Section 5.2.2.4.2).

Return_Port shall indicate the number of the port on the device which was used to respond for this indication (see Section 5.2.2.4.3).

Operational_Status shall contain the current operational status code for the device. Zero shall indicate that the device is fully operational; other codes are vendor specific for the device (see Section 5.2.2.4.3).

5.2.5.9 DIDS_READ_CAPABILITY_LIST.request

The DIDS_READ_CAPABILITY_LIST.request primitive shall permit an initiator to request the list of supported capabilities for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

5.2.5.10 DIDS_READ_CAPABILITY_LIST.indication

The DIDS_READ_CAPABILITY_LIST.indication primitive shall provide an indication to an initiator that the list of supported capabilities for a device may be available in response to a DIDS_READ_CAPABILITY_LIST.request. This primitive shall indicate whether the request was successful or not, and may contain a capability list.

The parameters for this primitive shall be:

- Result
- Capability_List

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Capability_List parameter may be invalid.

Capability_List shall contain the list of capabilities supported by the device in the form of supported and transport protocol IDs (see Section 5.2.2.5).

5.2.6 Initiator Primitive Provision

Initiators shall provide primitives in accordance with the rules shown in Table 5-7. “M” indicates that provision of the primitive is mandatory; “O” indicates that provision of the primitive is optional. Where an optional request primitive is provided, an indication primitive shall also be provided, and vice-versa.

Table 5-7: Device Identification Service Initiator Primitive Provision		
Primitive	**Provision**	**Notes**
DIDS_READ_INFO.request	M	-
DIDS_READ_INFO.indication	M	-
DIDS_READ_VENDOR_STRING.request	O	-
DIDS_READ_VENDOR_STRING.indication	O	-
DIDS_READ_PRODUCT_STRING.request	O	-
DIDS_READ_PRODUCT_STRING.indication	O	-
DIDS_READ_STATUS.request	M	-
DIDS_READ_STATUS.indication	M	-
DIDS_READ_CAPABILITY_LIST.request	O	-
DIDS_READ_CAPABILITY_LIST.indication	O	-

5.3 Network Management Service

5.3.1 Service Provision

Provision of the Network Management service is mandatory for both targets and initiators.

5.3.2 Target Parameters

Network Management parameters are summarised in Table 5-8.

Table 5-8: Network Management Service Parameters			
ID	**Name**	**Type**	**Summary**
0	Network Identification	Simple	Information for network discovery and a logical address
1-7	*Reserved*	-	*Reserved for future use*

5.3.2.1 Network Identification

Network Identification is a simple parameter comprised of the fields summarised in Table 5-9. Fields in the Network Identification parameter shall be read-write.

Table 5-9: Network Identification Parameter Fields		
ID	**Name**	**Summary**
0	Network ID	Permits an identifier to be assigned to this device
1	Device Logical Address	Logical address for this device
2-31	*Reserved*	*Reserved for future use*

5.3.2.1.1 Network ID Field

The Network ID field shall be a 32-bit read-write field which retains any value written to it. The Network ID field exists to permit network discovery.

5.3.2.1.2 Device Logical Address Field

The Device Logical Address field may be provided to permit the owner to provide the device with a logical address that may be used to route packets to the device. For example, this permits the device to specify a source logical address for initiating RMAP commands. If this field is provided, the Logical Address part shall be both readable and writable. If this field is not provided, the Logical Address part shall be read-only and shall have a value of zero. The Device Logical Address field shall be encoded as shown in Figure 5-8.

Figure 5-8: Device Logical Address Field Encoding

Bits 0 to 7 (where 0 is the least significant) may be provided to contain a logical address. Valid values for this part are 32 to 254. If an attempt is made to write an invalid value, the current value shall not be affected.

Bits 8 to 31 shall be reserved for future use; these bits shall be written as zero and shall be ignored when read.

5.3.3 Target Parameter Provision

Device targets shall provide parameters in accordance with the rules shown in Table 5-10. “M” indicates that provision of the parameter/field for the identified operation is mandatory; “O” indicates that provision of the parameter/field for the identified operation is optional.

Table 5-10: Network Management Service Target Parameter Provision				
Parameter	**Field(s)**	**Read**	**Write**	**RMW**
Network Identification	*All*	M	M	O

5.3.4 Target Actions

A target shall not carry out any further actions beyond support for read/write/read-modify-write commands specified in Section 5.3.3.

5.3.5 Initiator Primitives

A summary of Network Management service initiator primitives is presented below:

- NMS_READ_NETWORK_ID.request
- NMS_READ_NETWORK_ID.indication
- NMS_WRITE_NETWORK_ID.request
- NMS_WRITE_NETWORK_ID.indication
- NMS_READ_DEVICE_LA.request
- NMS_READ_DEVICE_LA.indication
- NMS_WRITE_DEVICE_LA.request
- NMS_WRITE_DEVICE_LA.indication
- NMS_DISCOVER_NETWORK.request
- NMS_DISCOVER_NETWORK.indication

These primitives are considered in more detail in the following sections.

5.3.5.1 NMS_READ_NETWORK_ID.request

The NMS_READ_NETWORK_ID.request primitive shall permit an initiator to request the current network ID for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

5.3.5.2 NMS_READ_NETWORK_ID.indication

The NMS_READ_NETWORK_ID.indication primitive shall provide an indication to an initiator that a network ID may be available in response to a NMS_READ_NETWORK_ID.request. This primitive shall indicate whether the request was successful or not, and may contain a network ID.

The parameters for this primitive shall be:

- Result
- Network_ID

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Network_ID parameter may be invalid.

Network_ID shall contain the current network ID of the device (see Section 5.3.2.1.1).

5.3.5.3 NMS_WRITE_NETWORK_ID.request

The NMS_WRITE_NETWORK_ID.request primitive shall permit an initiator to set the current network ID for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.
- Network_ID

RMAP_Parameters includes the source address, destination address and transaction ID.

Network_ID shall contain the desired network ID for the device (see Section 5.3.2.1.1).

5.3.5.4 NMS_WRITE_NETWORK_ID.indication

The NMS_WRITE_NETWORK_ID.indication primitive shall provide an indication to an initiator whether or not a network ID was written successfully in response to a NMS_WRITE_NETWORK_ID.request.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

5.3.5.5 NMS_READ_DEVICE_LA.request

The NMS_READ_DEVICE_LA.request primitive shall permit an initiator to request the current logical address for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

5.3.5.6 NMS_READ_DEVICE_LA.indication

The NMS_READ_DEVICE_LA.indication primitive shall provide an indication to an initiator that a device logical address may be available in response to a NMS_READ_DEVICE_LA.request. This primitive shall indicate whether the request was successful or not, and may contain a device logical address.

The parameters for this primitive shall be:

- Result
- Device_LA

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Logical_Address parameter may be invalid.

Device_LA shall contain the current logical address of the device (see Section 5.3.2.1.2).

5.3.5.7 NMS_WRITE_DEVICE_LA.request

The NMS_WRITE_DEVICE_LA.request primitive shall permit an initiator to set the current device logical address for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.
- Device_LA

RMAP_Parameters includes the source address, destination address and transaction ID.

Device_LA shall contain the desired logical address for the device (see Section 5.3.2.1.2).

5.3.5.8 NMS_WRITE_DEVICE_LA.indication

The NMS_WRITE_DEVICE_LA.indication primitive shall provide an indication to an initiator whether or not a device logical address was written successfully in response to a NMS_WRITE_DEVICE_LA.request.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

5.3.5.9 NMS_DISCOVER_NETWORK.request

The NMS_DISCOVER_NETWORK.request primitive shall initiate a breadth-first traversal of the network, gathering basic device and topology information which shall be supplied to the initiator as part of the NMS_DISCOVER_NETWORK.indication primitive.

As each device is discovered by this primitive it shall be uniquely identified in one of two ways:

1. By utilising the static device identifier accessible via the Device Identification service (if provided).
2. By assigning a unique network ID to the device.

The unique identification shall be used to detect loops in the network topology.

The parameters for this primitive shall be:

- Size_Limit

Size_Limit specifies a limit (as a number of devices) for the network traversal. It shall be possible to specify a value for this parameter such that no limit is placed on the network traversal.

5.3.5.10 NMS_DISCOVER_NETWORK.indication

The NMS_DISCOVER_NETWORK.indication primitive shall indicate to an initiator that a network topology may be available in response to a NMS_DISCOVER_NETWORK.request. This primitive shall indicate whether the request was successful or not, and may contain a device connectivity list identifying network devices and their connectivity and, by extension, the network topology.

The parameters for this primitive shall be:

- Result
- Device_Connectivity

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Device_Connectivity parameter may be invalid.

Device_Connectivity shall contain an enumerated list of devices discovered on the network, together with their connectivity. Each entry in the list describes a single network device and shall contain the following parameters:

- Vendor_ID
- Product_ID
- Preferred_Region
- Router_Node
- Support_Level
- Port_Count
- Device_ID
- Version
- Instance_ID
- Network_ID
- Link_List

Vendor_ID shall contain the vendor ID of the device (see Section 5.2.2.1.1).

Product_ID shall contain the product ID of the device (see Section 5.2.2.1.1).

Preferred_Region shall contain the preferred region identifier for the device; this may be zero if no region is preferred (see Section 5.2.2.1.2).

Router_Node shall indicate whether the device is a router or a node (see Section 5.2.2.1.2).

Support_Level shall indicate whether this device supports SpaceWire-PnP level 1 or 2 (see Section 5.2.2.1.2).

Port_Count shall contain the number of physical ports on the device (see Section 5.2.2.1.2).

Device_ID shall contain the device ID; this may be zero if no device ID is provided (see Section 5.2.2.1.3).

Version shall contain the version information for the device (see Section 5.2.2.1.4).

Instance_ID shall contain the instance ID for the device (see Section 5.2.2.1.4).

Network_ID shall contain the current network ID of the device (see Section 5.3.2.1.1).

Link_List shall contain an enumerated list, where each entry represents a physical port on this device. There shall be an entry for every physical port. Each entry shall have the following parameters:

- Connected_Device
- Connected_Port

Connected_Device shall reference an entry in the Device_Connectivity list indicating the device to which the physical port represented by this entry is connected. If the physical port represented by this entry is not connected, this parameter shall hold a distinct value (such as zero) to indicate this. Additionally, if the physical port represented by this entry is connected to a device not included in the Device_Connectivity list, this parameter shall hold a distinct value (such as -1) to indicate this.

Connected_Port shall indicate the physical port number on the device to which the physical port represented by this entry is connected (i.e. the physical port number on the remote device). If The Connected_Device parameter indicates that the physical port represented by this entry is either not connected, or is connected to a device not included in the Device_Connectivity list, this parameter may not be valid.

5.3.6 Initiator Primitive Provision

Initiators shall provide primitives in accordance with the rules shown in Table 5-11. “M” indicates that provision of the primitive is mandatory; “O” indicates that provision of the primitive is optional. Where an optional request primitive is provided, an indication primitive shall also be provided, and vice-versa.

Table 5-11: Network Management Service Initiator Primitive Provision		
Primitive	**Provision**	**Notes**
NMS_READ_NETWORK_ID.request	M	-
NMS_READ_NETWORK_ID.indication	M	-
NMS_WRITE_NETWORK_ID.request	M	-
NMS_WRITE_NETWORK_ID.indication	M	-
NMS_READ_DEVICE_LA.request	M	-
NMS_READ_DEVICE_LA.indication	M	-
NMS_WRITE_DEVICE_LA.request	M	-
NMS_WRITE_DEVICE_LA.indication	M	-
NMS_DISCOVER_NETWORK.request	O	-
NMS_DISCOVER_NETWORK.indication	O	-

5.4 Link Configuration Service

5.4.1 Service Provision

Provision of the Link Configuration service shall be mandatory for both targets and initiators.

5.4.2 Target Parameters

Link Configuration parameters are summarised in Table 5-12.

Table 5-12: Link Configuration Service Parameters			
ID	**Name**	**Type**	**Summary**
0	Port Activity	Simple	Specifies current and recent activity on all ports
1	Reference Rate	Simple	Permits control of the device reference transmit rate
2	Link Control	Complex	Permits link status reporting and configuration
3-7	*Reserved*	-	*Reserved for future use*

5.4.2.1 Port Activity

Port Activity is a simple parameter comprised of the fields summarised in Table 5-13. All fields in this parameter shall be read only.

Table 5-13: Port Activity Parameter Fields		
ID	**Name**	**Summary**
0	Active Ports	Bitmap of active ports
1	Active Ports Delta	Bitmap indicating changes in port activity
2-31	*Reserved*	*Reserved for future use*

5.4.2.1.1 Active Ports Field

The active ports field shall contain a bitmap indicating which device ports are in the Run state (see [AD1] for the SpaceWire link state machine). The field shall be encoded as follows:

- Each bit shall indicate whether the corresponding port is in the run state, i.e. bit 1 refers to port 1, bit 2 refers to port 2. A bit shall be 1 if the corresponding port is in the run state and 0 otherwise.
- Bits corresponding to external ports and bit 0 (corresponding to the configuration port) shall always be 1.
- Bits corresponding to ports which are not present on the device shall always be 0.

5.4.2.1.2 Active Ports Delta Field

The active ports delta field shall indicate which bits in the Active Ports field (see Section 5.2.2.4.2) have changed state at least once since the last time this field was read. A 1 in any bit position shall indicate that the corresponding bit in the Active Ports field has changed state at least once. Reading the field shall clear all bits to 0. The reset state of this field shall be zero.

A device may choose not to support the Active Ports Delta action in which case this field shall still be present and shall contain the value 0xFFFFFFFF.

5.4.2.2 Reference Rate

Reference Rate is a simple parameter comprised of the fields summarised in Table 5-14.

Table 5-14: Reference Rate Parameter Fields		
ID	**Name**	**Summary**
0	Base Rate/Mode	Base rate for the reference transmit rate and mode
1	Maximum/ Minimum Reference Rate	Maximum and minimum reference rate values
2	Reference Rate	Current setting for the reference transmit rate
3-31	*Reserved*	*Reserved for future use*

5.4.2.2.1 Base Rate/Mode Field

The Base Rate/Mode field shall specify the base value for the device reference rate in Mbps as well as the mode used for configuring the reference rate (rate value or divider). The reference transmit rate and all link transmit rates depend on the base reference rate. The Base Rate/Mode field shall be read-only and constant. Encoding of the Base Rate/Mode field is shown in Figure 5-9.

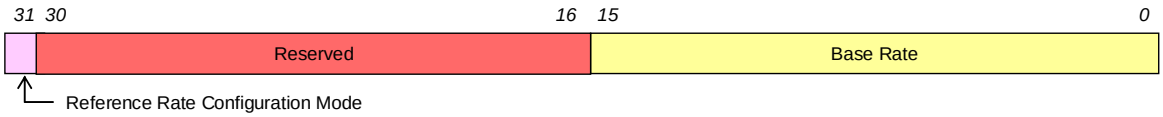


Figure 5-9: Base Rate/Mode Field Encoding

The field shall have 2 parts:

- Bit 31 shall indicate the mode that shall be used for specifying the reference rate (derived from the base rate). If this bit is set to 0 values in Mbps shall be used for specifying the reference rate; if this bit is set to 1 integer divider values shall be used for specifying the reference rate.
- Bits 0 to 15 (where 0 is the least significant) shall specify the base transmit rate for this device in Mbps. If the Reference Rate Configuration Mode (bit 31) is 0 (specifying rate values in Mbps) this value shall be the same as the Maximum Reference Rate value contained in the Maximum/ Minimum Reference Rate field (see Section 5.4.2.2.2).

Bits 16 to 30 shall be reserved for future use. The value of these bits shall be ignored when read.

5.4.2.2.2 Maximum/Minimum Reference Rate Field

The Maximum/Minimum Reference Rate field shall specify the maximum and minimum values permitted for the Reference Rate field. This field shall be read-only and constant. The encoding for the Maximum/ Minimum Reference Rate field is shown in Figure 5-10.

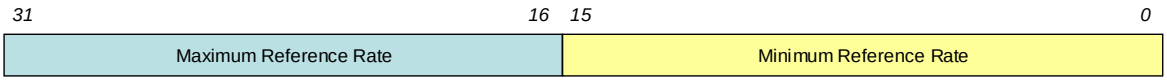


Figure 5-10: Maximum/ Minimum Reference Rate Field Encoding

The Maximum/Minimum Reference Rate field shall be split into two parts:

- Bits 16 to 31 (where 16 is the least significant) shall specify the maximum configuration value for the Reference Rate field. This shall be the numerically maximum value for that field, i.e. if the Reference Rate is set as a rate value (the Reference Rate Configuration Mode bit in the Base Rate/Mode field is zero) then this shall hold the maximum rate value that may be written to the Reference Rate field; if the Reference Rate is set as a divider (the Reference Rate Configuration Mode bit is one) then this shall hold the numerically maximum divider value that may be written to the Reference Rate field.
- Bits 0 to 15 (where 0 is the least significant) shall specify the minimum configuration value for the Reference Rate field. This shall be the numerically minimum value for that field, i.e. if the Reference Rate is set as a rate value (the Reference Rate Configuration Mode bit in the Base Rate/Mode field is zero) then this shall hold the minimum rate value that may be written to the Reference Rate field; if the Reference Rate is set as a divider (the Reference Rate Configuration Mode bit is one) then this shall hold the numerically minimum divider value that may be written to the Reference Rate field.

If the Reference Rate Configuration Mode bit in the Base Rate/Mode field is zero the maximum reference rate value shall be the same as the base rate value in the Base Rate/Mode field. If the reference rate is not used, both parts of this field shall be zero.

5.4.2.2.3 Reference Rate Field

The Reference Rate field controls the reference rate from which all link transmit rates are derived. If the Reference Rate Configuration Mode bit in the Base Rate/Mode field is zero, the value in this field shall specify the reference rate in Mbps. If the Reference Rate Configuration Mode bit in the Base Rate/Mode field is one, the value in this field shall specify an integer divider value by which the base rate shall be divided to achieve the reference rate. In both cases, the maximum and minimum valid values for this field shall be specified by the Maximum/Minimum Reference Rate field. This field shall be read-write; the default value shall be vendor specific.

The Reference Rate field shall be encoded as shown in Figure 5-10.

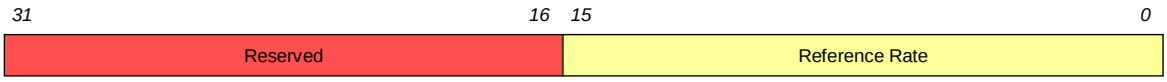


Figure 5-11: Reference Rate Field Encoding

Bits 0 to 15 (where 0 is the least significant) shall contain the reference rate. Bits 16 to 31 shall be reserved for future use. These bits shall be written as zero and shall be ignored when read.

The Reference Rate field may not support all values between the minimum and maximum. Values written to the Reference Rate field shall be handled as follows:

- If a value lower than the minimum value is written, the reference rate shall be set to the minimum value.
- If a value higher than the maximum value is written, the reference rate shall be set to the maximum value.
- If a value between the minimum and maximum is written, but that value is not supported, the reference rate shall be set to the highest supported value below the written value.

5.4.2.3 Link Control

Link Control is a complex parameter providing information about individual links and permitting their control.

Fields in the root entry shall be as shown in Table 5-15.

Table 5-15: Link Control Parameter Root Entry Fields

ID	Name	Summary
0	Link Count	Count of links (ports) on this device
1-31	*Reserved*	*Reserved for future use*

Non-root entries each represent a single link and shall have fields as shown in Table 5-16. There shall be as many non-root entries as there are ports on the device (not including the configuration port). Entry 1 shall represent port 1; entry 2 shall represent port 2 and so on. Valid entries shall be contiguous.

Table 5-16: Link Control Parameter Non-Root Entry Fields		
ID	**Name**	**Summary**
0	Link Type/Status	Link type and status
1	Maximum/Minimum Link Rate	Maximum and minimum link rate values
2	Link Rate/Priority/State	Link transmit rate, arbitration priority and state
3-31	*Reserved*	*Reserved for future use*

5.4.2.3.1 Link Count Field

The Link Count field specifies the number of valid entries in the Link Control parameter. The count shall be represented as an unsigned number from 1 to 31 (for nodes) or 2 to 31 (for routers), with the least significant bit being bit 0. The field encoding is shown in Figure 5-12. Bits 5 to 31 are reserved. The value of these bits shall be ignored. These bits are reserved for future use. The value of this field shall be identical to the number of ports given by the Number of Ports/Region field of the Device Information parameter provided by the Device Identification service. This field shall be read-only and constant.

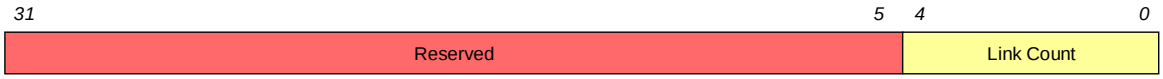


Figure 5-12: Link Count Field Encoding

5.4.2.3.2 Link Type/Status Field

The Link Type/Status field specifies the type of this link (SpaceWire/external), the mode used for setting the link rate, and the current link status. The Link Type/Status field shall be encoded as shown in Figure 5-13.

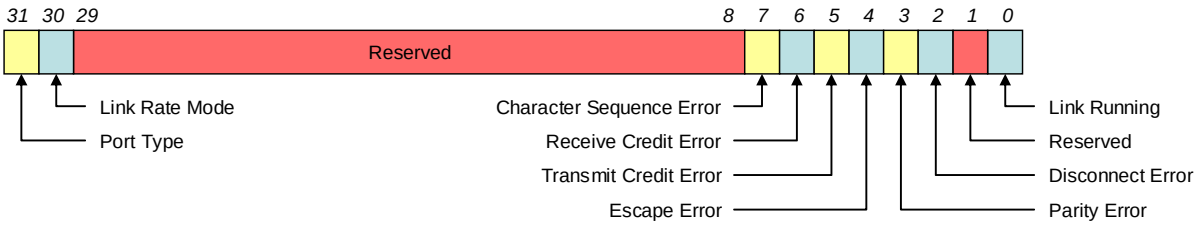


Figure 5-13: Link Type/Status Field Encoding

Bits in the Link Type/Status field shall represent the following:

- Bit 31 shall represent the port type. This bit shall be set to zero if the corresponding port is an external port; this bit shall be set to one if the corresponding port is a SpaceWire port.
- Bit 30 shall represent the link rate mode used by this link. If this bit is set to 0 values in Mbps shall be used for specifying the link rate; if this bit is set to 1 integer divider values shall be used for specifying the link rate.
- Bit 7 shall be set to one if a character sequence error has occurred on this link since the errors were cleared. The default value of this bit shall be zero.
- Bit 6 shall be set to one if a receive credit error has occurred on this link since the errors were cleared. The default value of this bit shall be zero.
- Bit 5 shall be set to one if a transmit credit error has occurred on this link since the errors were cleared. The default value of this bit shall be zero.
- Bit 4 shall be set to one if an escape error has occurred on this link since the errors were cleared. The default value of this bit shall be zero.
- Bit 3 shall be set to one if a parity error has occurred on this link since the errors were cleared. The default value of this bit shall be zero.
- Bit 2 shall be set to one if a disconnect error has occurred on this link since the errors were cleared. The default value of this bit shall be zero.
- Bit 0 shall be set to one if the link is in the run state. This bit shall always be one for external ports.

Bit 1 and bits 8 to 29 are reserved. The value of these bits shall be ignored. These bits are reserved for future use.

To clear errors, the value zero should be written to this field. No other value should be written; all other values are reserved for future use.

5.4.2.3.3 Maximum/Minimum Link Rate Field

The Maximum/Minimum Link Rate field shall specify the maximum and minimum values permitted for the Link Rate field. This field shall be read-only although it may not be constant: the values in this field may depend on the value of the Reference Rate field of the Reference Rate parameter. The encoding for the Maximum/Minimum Link Rate field is shown in Figure 5-14.

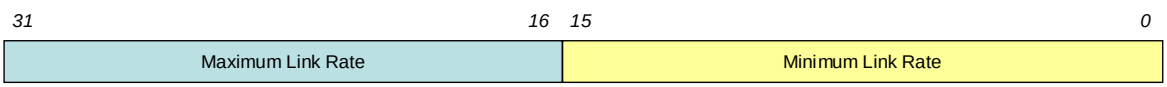


Figure 5-14: Maximum/ Minimum Link Rate Field Encoding

The Maximum/Minimum Link Rate field shall be split into two parts:

- Bits 16 to 31 (where 16 is the least significant) shall specify the maximum configuration value for the Link Rate field. This shall be the numerically maximum value for that field, i.e. if the Link Rate is set as a rate value (the Link Rate Mode bit in the Link Type/Status field is zero) then this shall hold the maximum rate value that may be written to the Link Rate field; if the Link Rate is set as a divider (the Link Rate Mode bit is one) then this shall hold the numerically maximum divider value that may be written to the Link Rate field.
- Bits 0 to 15 (where 0 is the least significant) shall specify the minimum configuration value for the Link Rate field. This shall be the numerically minimum value for that field, i.e. if the Link Rate is set as a rate value (the Link Rate Mode bit in the Link Type/Status field is zero) then this shall hold the minimum rate value that may be written to the Link Rate field; if the Link Rate is set as a divider (the Link Rate Configuration Mode bit is one) then this shall hold the numerically minimum divider value that may be written to the Link Rate field.

5.4.2.3.4 Link Rate/Priority/State Field

The Link Rate/Priority/State field shall permit the control of the link transmit rate, the arbitration priority for routing and the current link state. The Link Rate/Priority/State field shall be encoded as shown in Figure 5-15.

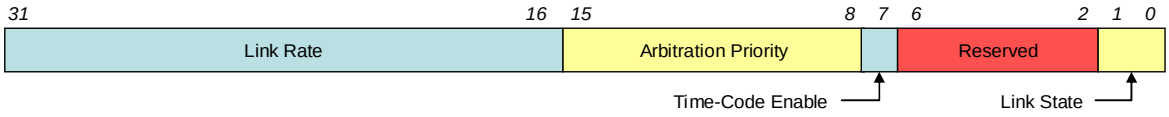


Figure 5-15: Link Rate/Priority/State Field Encoding

The Link Rate/Priority/State field shall have four parts:

- Bits 16 to 31 (where 16 is the least significant) shall contain the link rate. If the Link Rate Mode bit in the Link Type/Status field is zero, the value shall specify the link rate in Mbps. If the Link Rate Mode bit in the Link Type/Status field is one, the value here shall specify an integer divider value by which the reference rate shall be divided to achieve the link rate. In both cases, the maximum and minimum valid values for the link rate shall be specified by the Maximum/Minimum Link Rate field. The default value of the link rate shall be vendor specific.
- On routers, bits 8 to 15 (where 8 is the least significant) shall contain the arbitration priority for use in routing. The numeric value of this field shall indicate priority, where 0 is the lowest, and 255 is the highest. Vendors may choose the number of priority levels to permit, from 0 to

256. Valid priority levels shall always be contiguous and the default priority level shall be 0. If a priority value is written which is higher than the highest supported value, the priority value shall be set to the highest supported value. On node devices bits 8 to 15 shall be reserved for future use. These bits shall be written as zero and shall be ignored when read.

- Bit 7 shall enable time-code propagation (transmission) through this link. If set to 1, time-code transmission is enabled, if set to 0, time-code transmission is disabled.
- Bits 0 and 1 shall control the state of a link between idle, started, autostart and disabled. Started and autostart shall be implemented as defined in the SpaceWire standard [AD1]. Idle and disabled states shall both prevent a link from starting if the link is not currently in the run state, if the link is currently running, idle shall not cause the link to leave the run state, whereas disabled shall cause the link to leave the run state. States are encoded as shown in Table 5-17. SpaceWire ports shall always implement all states. External ports shall either always be started, or shall be selectable as started or disabled, i.e. for external ports bit 0 shall always be 1 and bit 1 shall either be read-only and set to 0 or shall be writable.

Table 5-17: Link State Encoding

Bit 1	Bit 0	State
0	0	Idle
0	1	Started
1	0	Autostart
1	1	Disabled

The link rate may not support all values between the minimum and maximum. Values written to the link rate shall be handled as follows:

- If a value lower than the minimum value is written, the link rate shall be set to the minimum value.
- If a value higher than the maximum value is written, the link rate shall be set to the maximum value.
- If a value between the minimum and maximum is written, but that value is not supported, the link rate shall be set to the highest supported value below the written value.

Bits 2 to 6 shall be reserved for future use. These bits shall be written as zero and shall be ignored when read.

5.4.3 Target Parameter Provision

Device targets shall provide parameters in accordance with the rules shown in Table 5-18. "M" indicates that provision of the parameter/field for the identified operation is mandatory; "O" indicates that provision of the parameter/field for the identified operation is optional; "F" indicates that provision of the parameter/field for the identified operation is forbidden.

Table 5-18: Link Configuration Service Target Parameter Provision

Parameter	Field(s)	Read	Write	RMW
Port Activity	*All*	M	F	F
Reference Rate	Base Rate/Mode	M*	F	F
Reference Rate	Maximum/Minimum Reference Rate	M*	F	F
Reference Rate	Reference Rate	M	M	O
Link Control	*Root entry fields*	M*	F	F
Link Control	Link Type/Status	M	F	F
Link Control	Maximum/Minimum Link Rate	M	M	O
Link Control	Link Rate/Priority/State	M	M	O

* All field values shall be constant.

5.4.4 Target Actions

Targets may carry out one action beyond support for read/write/read-modify-write commands specified in Section 5.4.3.

5.4.4.1 Active Ports Delta Update

The Active Ports Delta field shall be updated each time the Active Ports field changes; if a bit in the the Active Ports field changes the corresponding bit in the Active Ports Delta field shall be set.

When the Active Ports Delta field is read, it shall be reset to a value of zero.

If this action is not supported, the Active Ports Delta field shall be constant and shall contain a value of 0xFFFFFFFF.

5.4.5 Initiator Primitives

A summary of Link Configuration service initiator primitives is presented below:

- LCS_READ_PORT_ACTIVITY.request

- LCS_READ_PORT_ACTIVITY.indication
- LCS_READ_REFERENCE_RATE.request
- LCS_READ_REFERENCE_RATE.indication
- LCS_WRITE_REFERENCE_RATE.request
- LCS_WRITE_REFERENCE_RATE.indication
- LCS_READ_LINK_CONTROL.request
- LCS_READ_LINK_CONTROL.indication
- LCS_WRITE_LINK_RATE.request
- LCS_WRITE_LINK_RATE.indication
- LCS_WRITE_LINK_PRIORITY.request
- LCS_WRITE_LINK_PRIORITY.indication
- LCS_WRITE_LINK_STATE.request
- LCS_WRITE_LINK_STATE.indication

These primitives are considered in more detail in the following sections.

5.4.5.1 LCS_READ_PORT_ACTIVITY.request

The LCS_READ_PORT_ACTIVITY.request primitive shall permit an initiator to request the port activity information for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

5.4.5.2 LCS_READ_PORT_ACTIVITY.indication

The LCS_READ_PORT_ACTIVITY.indication primitive shall provide an indication to an initiator that port activity information may be available in response to a LCS_READ_PORT_ACTIVITY.request. This primitive shall indicate whether the request was successful or not, and may contain port activity information.

The parameters for this primitive shall be:

- Result
- Active_Ports

- Active_Ports_Delta

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then other parameters may be invalid.

Active_Ports shall contain a list of currently active ports on the device (see Section 5.4.2.1.1).

Active_Ports_Delta shall contain the list of ports which have undergone a change in activity status since the last time the field was read (see Section 5.4.2.1.2).

5.4.5.3 LCS_READ_REFERENCE_RATE.request

The LCS_READ_REFERENCE_RATE.request primitive shall permit an initiator to request the reference rate information for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

5.4.5.4 LCS_READ_REFERENCE_RATE.indication

The LCS_READ_REFERENCE_RATE.indication primitive shall provide an indication to an initiator that reference rate information may be available in response to a LCS_READ_REFERENCE_RATE.request. This primitive shall indicate whether the request was successful or not, and may contain reference rate information.

The parameters for this primitive shall be:

- Result
- Rate_Mode
- Base_Rate
- Maximum_Rate
- Minimum_Rate
- Current_Rate

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then other parameters may be invalid.

Rate_Mode shall indicate whether the reference rate is specified as a rate (in Mbps) or a divider of the base rate (see Section 5.4.2.2.1).

Base_Rate shall contain the base rate for this device in Mbps (see Section 5.4.2.2.1).

Maximum_Rate shall contain the maximum reference rate setting for this device; this will either be a rate (in Mbps) or a divider value depending on the value of Rate_Mode (see Section 5.4.2.2.2).

Minimum_Rate shall contain the minimum reference rate setting for this device; this will either be a rate (in Mbps) or a divider value depending on the value of Rate_Mode (see Section 5.4.2.2.2).

Current_Rate shall contain the current reference rate setting for this device; this will either be a rate (in Mbps) or a divider value depending on the value of Rate_Mode (see Section 5.4.2.2.3).

5.4.5.5 LCS_WRITE_REFERENCE_RATE.request

The LCS_WRITE_REFERENCE_RATE.request primitive shall permit an initiator to set the current reference rate for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.
- Current_Rate

RMAP_Parameters includes the source address, destination address and transaction ID.

Current_Rate shall contain the desired reference rate setting for this device; this shall either be a rate (in Mbps) or a divider value depending on the value of the Rate_Mode parameter of the LCS_READ_REFERENCE_RATE.indication primitive (see Section 5.4.5.4).

5.4.5.6 LCS_WRITE_REFERENCE_RATE.indication

The LCS_WRITE_REFERENCE_RATE.indication primitive shall provide an indication to an initiator whether or not a reference rate was written successfully in response to a LCS_WRITE_REFERENCE_RATE.request.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

5.4.5.7 LCS_READ_LINK_CONTROL.request

The LCS_READ_LINK_CONTROL.request primitive shall permit an initiator to request the link control information for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters;
- Link_Number

RMAP_Parameters includes the source address, destination address and transaction ID.

Link_Number shall indicate the number of the link for which link control information is being requested. This number shall be greater than zero and shall be equal to or less than the number of physical ports on the device.

5.4.5.8 LCS_READ_LINK_CONTROL.indication

The LCS_READ_LINK_CONTROL.indication primitive shall provide an indication to an initiator that link control information may be available in response to a LCS_READ_LINK_CONTROL.request. This primitive shall indicate whether the request was successful or not, and may contain link control information.

The parameters for this primitive shall be:

- Result
- Type
- Status
- Maximum_Rate
- Minimum_Rate
- Current_Rate
- Priority
- State

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then other parameters may be invalid.

Type shall indicate what type of port is present at the specified Link_Number; this will either be an external port or a SpaceWire port (see Section 5.4.2.3.2).

Status shall contain the current error status of the link (see Section 5.4.2.3.2)

Rate_Mode shall indicate whether the link rate is specified as a rate (in Mbps) or a divider of the current reference rate (see Section 5.4.2.3.2).

Maximum_Rate shall contain the maximum link rate setting for this link; this will either be a rate (in Mbps) or a divider value depending on the value of Rate_Mode (see Section 5.4.2.3.3).

Minimum_Rate shall contain the minimum link rate setting for this link; this will either be a rate (in Mbps) or a divider value depending on the value of Rate_Mode (see Section 5.4.2.3.3).

Current_Rate shall contain the current link rate setting for this link; this will either be a rate (in Mbps) or a divider value depending on the value of Rate_Mode (see Section 5.4.2.3.4).

Priority shall contain the current priority value for this link as an integer (see Section 5.4.2.3.4).

State shall contain the current state for this link which shall have four values representing each of the possible link states: Idle, Started, Autostart, Disabled (see Section 5.4.2.3.4).

5.4.5.9 LCS_WRITE_LINK_RATE.request

The LCS_WRITE_LINK_RATE.request primitive shall permit an initiator to set the link rate for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters
- Link_Number
- Current_Rate

RMAP_Parameters includes the source address, destination address and transaction ID.

Link_Number shall indicate the number of the link for which link control information should be written. This number shall be greater than zero and shall be equal to or less than the number of physical ports on the device.

Current_Rate shall contain the current link rate setting for this link (see Section 5.4.2.3.4); this will either be a rate (in Mbps) or a divider value depending on the value of the Rate_Mode parameter of the LCS_READ_LINK_CONTROL.indication primitive.

5.4.5.10 LCS_WRITE_LINK_RATE.indication

The LCS_WRITE_LINK_RATE.indication primitive shall provide an indication to an initiator whether or not link rate information was written successfully in response to a LCS_WRITE_LINK_RATE.request.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

5.4.5.11 LCS_WRITE_LINK_PRIORITY.request

The LCS_WRITE_LINK_PRIORITY.request primitive shall permit an initiator to set the link arbitration priority for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters
- Link_Number
- Priority

RMAP_Parameters includes the source address, destination address and transaction ID.

Link_Number shall indicate the number of the link for which link control information should be written. This number shall be greater than zero and shall be equal to or less than the number of physical ports on the device.

Priority shall contain the desired priority value for this link as an integer (see Section 5.4.2.3.4).

5.4.5.12 LCS_WRITE_LINK_PRIORITY.indication

The LCS_WRITE_LINK_PRIORITY.indication primitive shall provide an indication to an initiator whether or not link arbitration priority was written successfully in response to a LCS_WRITE_LINK_PRIORITY.request.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

5.4.5.13 LCS_WRITE_LINK_STATE.request

The LCS_WRITE_LINK_CONTROL.request primitive shall permit an initiator to set the link state for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters
- Link_Number
- State

RMAP_Parameters includes the source address, destination address and transaction ID.

Link_Number shall indicate the number of the link for which link control information should be written. This number shall be greater than zero and shall be equal to or less than the number of physical ports on the device.

State shall contain the desired state for this link which shall be one of four possible values each representing one of the possible link states: Idle, Started, Autostart, Disabled (see Section 5.4.2.3.4).

5.4.5.14 LCS_WRITE_LINK_STATE.indication

The LCS_WRITE_LINK_CONTROL.indication primitive shall provide an indication to an initiator whether or not the link state was written successfully in response to a LCS_WRITE_LINK_STATE.request.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

5.4.6 Initiator Primitive Provision

Initiators shall provide primitives in accordance with the rules shown in Table 5-11. “M” indicates that provision of the primitive is mandatory; “O” indicates that provision of the primitive is optional. Where an optional request primitive is provided, an indication primitive shall also be provided, and vice-versa.

Table 5-19: Link Configuration Service Initiator Primitive Provision		
Primitive	**Provision**	**Notes**
LCS_READ_PORT_ACTIVITY.request	M	-
LCS_READ_PORT_ACTIVITY.indication	M	-
LCS_READ_REFERENCE_RATE.request	M	-
LCS_READ_REFERENCE_RATE.indication	M	-
LCS_WRITE_REFERENCE_RATE.request	M	-
LCS_WRITE_REFERENCE_RATE.indication	M	-
LCS_READ_LINK_CONTROL.request	M	-
LCS_READ_LINK_CONTROL.indication	M	-
LCS_WRITE_LINK_RATE.request	M	-
LCS_WRITE_LINK_RATE.indication	M	-
LCS_WRITE_LINK_PRIORITY.request	O	-
LCS_WRITE_LINK_PRIORITY.indication	O	-
LCS_WRITE_LINK_STATE.request	M	-
LCS_WRITE_LINK_STATE.indication	M	-

5.5 ROUTER CONFIGURATION SERVICE

5.5.1 Service Provision

Provision of the Router Configuration service shall be mandatory for all initiators and router targets only. Node targets shall not provide this service.

5.5.2 Target Parameters

Router Configuration parameters are summarised in Table 5-20.

Table 5-20: Router Configuration Service Parameters			
ID	**Name**	**Type**	**Summary**
0	Router Control	Simple	Permits control of the routing operation of this device
1	Routing Table	Complex	Configure port and logical address routing
2-7	*Reserved*	-	*Reserved for future use*

5.5.2.1 Router Control

Router Control is a simple parameter comprised of the fields summarised in Table 5-21.

Table 5-21: Router Control Parameter Fields		
ID	**Name**	**Summary**
0	Watchdog Timeout	Sets the non-blocking protection watchdog timer
1	Arbitration Mode	Permits configuration of the routing arbitration mechanism
2	Time-Code Counters	Permits control of up to 4 time-code counters
3-31	*Reserved*	*Reserved for future use*

5.5.2.1.1 Watchdog Timeout Field

The Watchdog Timeout field shall contain the current value of the non-blocking protection watchdog timer. This shall be a 32-bit value (where bit 0 is the least significant). A value of zero shall indicate immediate timeout; a value of 0xFFFFFFFF shall indicate an infinite timeout (watchdog timeout disabled). All other values shall indicate the period of the watchdog timeout in microseconds.

The Watchdog Timeout field shall be readable and writable. A device may chose not to support the watchdog timeout function, in which case this field shall be read-only, constant and shall contain 0xFFFFFFFF. If the watchdog feature is implemented, it may not support all possible values. Values written to the Watchdog Timeout field shall be handled as follows:

- If a value lower than the minimum supported value is written, the watchdog timeout shall be set to the minimum supported value.
- If a value higher than the maximum supported value is written, the watchdog timeout shall be set to the maximum supported value (infinity shall be treated distinctly).
- If a value between the minimum and maximum is written, but that value is not supported, the watchdog timeout shall be set to the lowest supported value above the written value.

5.5.2.1.2 Arbitration Mode Field

The Arbitration Mode field shall permit configuration of the mechanism used to arbitrate between incoming packets competing for an output port. The Arbitration Mode field shall be encoded as shown in Figure 5-16.

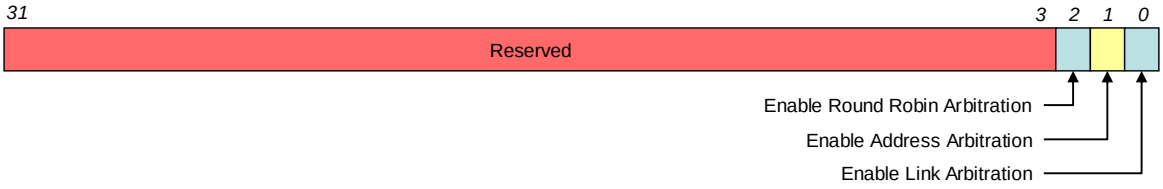


Figure 5-16: Arbitration Mode Field Encoding

The router arbitration mechanism shall be determined according to the value of bits 0 to 2 of the Arbitration Mode field. Each bit may be writable, if the device implements the arbitration mechanism associated with that bit. A device may choose to make any bit read-only if the arbitration mechanism associated with that bit cannot be enabled/disabled.

- If bit 0 is set to 1, link arbitration shall be used. This shall utilise the priority part of the Link Rate/Priority/State field of all non-root entries in the Link Control parameter (provided by the Link Configuration service) where entries are associated with links having packets waiting; the numerically highest priority value shall be transmitted first. If bit 0 is set to 0, link arbitration shall not be used.
- If bit 1 is set to 1, address arbitration shall be used. This shall utilise the priority part of the Address Control field of the Routing Table parameter, using the entries which relate to the addresses specified by the waiting packets. The packet with the address associated with the numerically highest priority value shall be transmitted first. If bit 1 is set to 0, address arbitration shall not be used.
- If bit 2 is set to 1, round robin arbitration shall be used. This shall ensure that each destination port gives priority to ports with packets waiting fairly, permitting transmission from each port in turn. If bit 2 is set to 0, round robin arbitration shall not be used.

Where more than one bit is set, the arbitration mechanisms shall be applied in the following order:

1. Link arbitration (if enabled)
2. Address arbitration (if enabled)
3. Round robin arbitration (if enabled)

If no arbitration mechanism is enabled, or enabled arbitration mechanisms have failed to result in packet transmission, the port number of links which have packets waiting shall be used to determine

which packet should be transmitted, with numerically lower link numbers having their packet transmitted first.

Bits 3 to 31 are reserved for future use. These bits shall be written as zero and shall be ignored when read.

5.5.2.1.3 Time-Code Counters Field

The Time-Code Counters field provides the current values of up to 4 time-code counters, each associated with a combination of time-code flags. Routers may choose to support from 1 to 4 time-code counters; at least one time-code counter shall be supported. The Time-Code Counters field shall be encoded as shown in Figure 5-17.

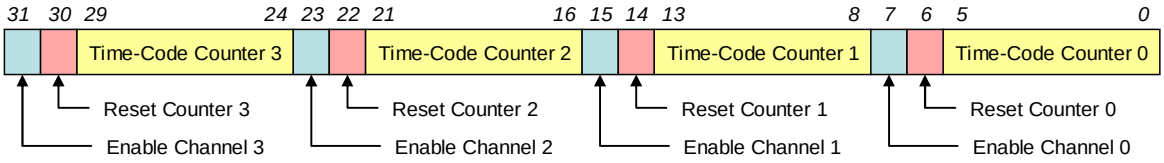


Figure 5-17: Time-Code Counters Field Encoding

The Time-Code Counters field has parts associated with each of the four possible time-code channels, where a channel refers to a combination of time-code flags in a time-code value. Channel 0 refers to time-code flags of 00; channel 1 refers to time-code flags of 01; channel 2 refers to time-code flags of 10; channel 3 refers time-code flags of 11.

- Bits 0 to 5 (where 0 is the least significant) shall contain the current time-code value for channel 0, and shall be read-only.
- Bit 6 shall always read as zero. Writing this bit as a 1 shall reset the time-code value for channel 0.
- Bit 7 shall control time-code channel 0, if this bit is set, time-code counter 0 shall be updated when time-codes for channel 0 arrive, and time-codes associated with channel shall be propagated according to the SpaceWire standard [AD1].
- Bits 8 to 13 (where 8 is the least significant) shall contain the current time-code value for channel 1, if it is supported, and shall be read-only.
- Bit 14 shall always read as zero. Writing this bit as a 1 shall reset the time-code value for channel 1, if it is supported.
- Bit 15 shall control time-code channel 1, if this bit is set, time-code counter 1 shall be updated when time-codes for channel 1 arrive, and time-codes associated with channel shall be

propagated according to the SpaceWire standard [AD1]. If time-code channel 1 is not supported, this bit shall always be zero.

- Bits 16 to 21 (where 16 is the least significant) shall contain the current time-code value for channel 2, if it is supported, and shall be read-only.
- Bit 22 shall always read as zero. Writing this bit as a 1 shall reset the time-code value for channel 2, if it is supported.
- Bit 23 shall control time-code channel 2, if this bit is set, time-code counter 2 shall be updated when time-codes for channel 2 arrive, and time-codes associated with channel shall be propagated according to the SpaceWire standard [AD1]. If time-code channel 2 is not supported, this bit shall always be zero.
- Bits 24 to 29 (where 24 is the least significant) shall contain the current time-code value for channel 3, if it is supported, and shall be read-only.
- Bit 30 shall always read as zero. Writing this bit as a 1 shall reset the time-code value for channel 3, if it is supported.
- Bit 31 shall control time-code channel 3, if this bit is set, time-code counter 3 shall be updated when time-codes for channel 3 arrive, and time-codes associated with channel shall be propagated according to the SpaceWire standard [AD1]. If time-code channel 3 is not supported, this bit shall always be zero.

5.5.2.2 Routing Table

Routing Table is a complex parameter permitting configuration of physical and logical address routing.

Fields in the root entry shall be as shown in Table 5-22.

Table 5-22: Routing Table Parameter Root Entry Fields

ID	Name	Summary
0	Logical Address Count	Count of logical addresses supported by this router
1-31	*Reserved*	*Reserved for future use*

The root entry contains a single field specifying the number of valid logical addresses supported by this router. Routers shall provide between 0 and 223 valid logical addresses.

Fields in non-root entries each represent a single address (either physical or logical) and shall be as shown in Table 5-23. There shall be at least as many valid non-root entries in the Routing Table as

there are physical ports. Entries from 1 to the value of the highest numbered physical port shall represent physical addresses. Entries higher than the valid physical addresses, but lower than 32 (if the highest physical address is less than 31), shall be reserved. These entries shall neither be read from nor written to. Entries from 32 to the highest supported logical address, which has a maximum value of 254, shall represent logical addresses. Logical addresses shall be provided contiguously starting from 32. Entries higher than the valid logical addresses shall be reserved. These entries shall neither be read from nor written to.

Table 5-23: Routing Table Parameter Non-Root Entry Fields

ID	Name	Summary
0	Port Association	Destination port(s) for routing this address
1	Address Control	Configuration of routing action for this address
2-31	*Reserved*	*Reserved for future use*

5.5.2.2.1 Logical Address Count Field

The Logical Address Count field shall identify the number of logical addresses supported by this router. The field shall be encoded as shown in Figure 5-18.

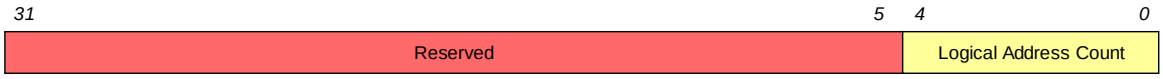


Figure 5-18: Logical Address Count Field Encoding

The value of the Logical Address Count field shall be specified as an integer in bits 0-7 (where 0 is the least significant). Values between 0 and 223 shall be permitted. This field shall be read-only and constant.

5.5.2.2.2 Port Association Field

The Port Association field shall indicate which ports are to be used as a destination for the address associated with this entry number. If a bit is set to 1, the port corresponding to that bit number shall be used as a destination port. Bit 0 shall be 0 and read-only. Bits not corresponding to physical ports (i.e. bit numbers above the highest physical port) shall be reserved; these bits shall be written as zero and ignored when read. For entries representing physical ports (non-reserved entries below 32), the bit in the Port Association field corresponding to the entry number shall be 1 and read-only. Additionally, for physical port entries, this field may be entirely read only. Where more than one bit is set, either group adaptive routing (GAR) or packet distribution (PD) will be used, according to the selection in the Address Control field. Routers may place limits on the number of bits that may be set

concurrently in the Port Association field. When a write value specifies a greater number of set bits than are supported by the router, the mechanism for determining which bits should remain set shall be vendor specific.

5.5.2.2.3 Address Control Field

The Address Control field permits configuration of the routing action of the address associated with this entry number. The Address Control field shall be encoded as shown in Figure 5-19.

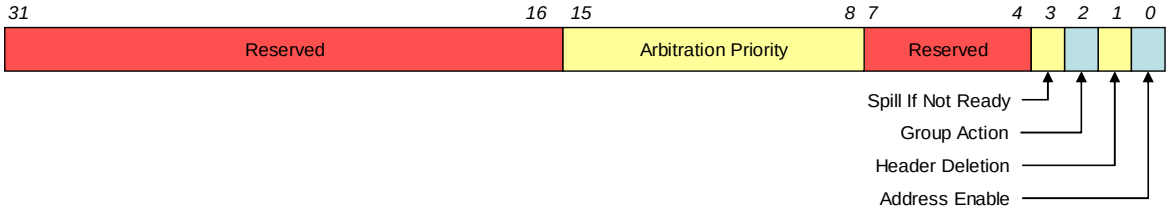


Figure 5-19: Address Control Field Encoding

The Address Control field shall have the following parts:

- Bit 0 shall be set to 1 to enable routing to this address. When this bit is set to 0, the router will recognise this address as invalid and shall discard all packets destined for this address. This bit shall default to 1 for all valid physical address entries and may be read-only; this bit shall always be writable and shall default to 0 for all valid logical addresses.
- Bit 1 shall be set to 1 to instruct the router to remove the address byte from packet header before routing the packet to its destination, when the packet carries the address associated with this entry. This bit shall be 1 and read-only for all valid physical address entries; this bit shall be writable and shall default to 0 for all valid logical addresses.
- Bit 2 shall indicate the action that the router will take when multiple bits are set to 1 in the Port Association field for this entry. If this bit is set to 0, the router shall choose output port with the lowest physical port number which a) has the corresponding bit set in the Port Association field; b) the link is in the run state; and c) the link is not busy. This is GAR. If this bit is set to 1, the router shall duplicate an output packet on all physical ports which a) have the corresponding bit set in the Port Association field; and b) the link is in the run state; waiting for all links to become available. This is PD. This bit may be read-only if either PD or GAR is not supported.
- Bit 3 shall be set to 1 to indicate to the router that incoming packets should be immediately spilt if: none of the potential output ports are ready (if bit 2 is set for GAR mode); any one of the potential output ports is not ready (if bit 2 is set for PD mode). If this bit is set to 0,

packets shall not be spilt by the router under these conditions. This bit shall default to 0 for all addresses and may be read-only and zero.

- Bits 8 to 15 (where 8 is the least significant) shall contain the arbitration priority for use in routing. The numeric value of this field shall indicate priority, where 0 is the lowest, and 255 is the highest. Vendors may choose the number of priority levels to permit, from 0 to 256. Valid priority levels shall always be contiguous and the default priority level shall be 0. If a priority value is written which is higher than the highest supported value, the priority value shall be set to the highest supported value. Routers must implement the same valid priority range for all logical addresses and the same valid priority range for all physical addresses (although this may be a different range to that supported for logical addresses).

Bits 4 to 7 and 16 to 31 shall be reserved for future use. These bits shall be written as 0 and ignored when read.

5.5.3 Target Parameter Provision

Device targets shall provide parameters in accordance with the rules shown in Table 5-24. "M" indicates that provision of the parameter/field for the identified operation is mandatory; "O" indicates that provision of the parameter/field for the identified operation is optional; "F" indicates that provision of the parameter/field for the identified operation is forbidden.

Table 5-24: Router Configuration Service Target Parameter Provision

Parameter	Field(s)	Read	Write	RMW
Router Control	*All*	M	M	O
Routing Table	*Root entry fields*	M*	F	F
Routing Table	*Non-root entry fields*	M	M	O

* All field values shall be constant.

5.5.4 Target Actions

A target shall not carry out any further actions beyond support for read/write/read-modify-write commands specified in Section 5.5.3.

5.5.5 Initiator Primitives

A summary of Router Configuration service initiator primitives is presented below:

- RCS_READ_WATCHDOG_TIMEOUT.request

- RCS_READ_WATCHDOG_TIMEOUT.indication
- RCS_WRITE_WATCHDOG_TIMEOUT.request
- RCS_WRITE_WATCHDOG_TIMEOUT.indication
- RCS_READ_ARBITRATION_MODE.request
- RCS_READ_ARBITRATION_MODE.request
- RCS_WRITE_ARBITRATION_MODE.request
- RCS_WRITE_ARBITRATION_MODE.request
- RCS_READ_TIME_COUTER.request
- RCS_READ_TIME_COUTER.indication
- RCS_RESET_TIME_COUTER.request
- RCS_RESET_TIME_COUTER.indication
- RCS_ENABLE_TIME_COUNTER.request
- RCS_ENABLE_TIME_COUNTER.indication
- RCS_READ_LA_COUNT.request
- RCS_READ_LA_COUNT.indication
- RCS_READ_ROUTING_TABLE_ENTRY.request
- RCS_READ_ROUTING_TABLE_ENTRY.indication
- RCS_WRITE_ROUTING_TABLE_ENTRY.request
- RCS_WRITE_ROUTING_TABLE_ENTRY.indication

These primitives are considered in more detail in the following sections.

5.5.5.1 RCS_READ_WATCHDOG_TIMEOUT.request

The RCS_READ_WATCHDOG_TIMEOUT.request primitive shall permit an initiator to request the current watchdog timeout value for any routing device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

5.5.5.2 RCS_READ_WATCHDOG_TIMEOUT.indication

The RCS_READ_WATCHDOG_TIMEOUT.indication primitive shall provide an indication to an initiator that a watchdog timeout value may be available in response to a RCS_READ_WATCHDOG_TIMEOUT.request. This primitive shall indicate whether the request was successful or not, and may contain a watchdog timeout value.

The parameters for this primitive shall be:

- Result
- Watchdog_Timeout

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Watchdog_Timeout parameter may be invalid.

Watchdog_Timeout shall contain the current watchdog timeout value of the router (see Section 5.5.2.1.1).

5.5.5.3 RCS_WRITE_WATCHDOG_TIMEOUT.request

The RCS_WRITE_WATCHDOG_TIMEOUT.request primitive shall permit an initiator to set the current watchdog timeout value for any routing device.

The parameters for this primitive shall be:

- RMAP_Parameters.
- Watchdog_TImeout

RMAP_Parameters includes the source address, destination address and transaction ID.

Watchdog_Timeout shall contain the desired watchdog timeout value of the router (see Section 5.5.2.1.1).

5.5.5.4 RCS_WRITE_WATCHDOG_TIMEOUT.indication

The RCS_WRITE_WATCHDOG_TIMEOUT.indication primitive shall provide an indication to an initiator whether or not a watchdog timeout value was written successfully in response to a RCS_WRITE_WATCHDOG_TIMEOUT.request.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

5.5.5.5 RCS_READ_ARBITRATION_MODE.request

The RCS_READ_ARBITRATION_MODE.request primitive shall permit an initiator to request the current arbitration mode for any routing device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

5.5.5.6 RCS_READ_ARBITRATION_MODE.indication

The RCS_READ_ARBITRATION_MODE.indication primitive shall provide an indication to an initiator that an arbitration mode value may be available in response to a RCS_READ_ARBITRATION_MODE.request. This primitive shall indicate whether the request was successful or not, and may contain an arbitration mode.

The parameters for this primitive shall be:

- Result
- Arbitration_Mode

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Arbitration_Mode parameter may be invalid.

Arbitration_Mode shall contain the current arbitration mode of the router (see Section 5.5.2.1.2).

5.5.5.7 RCS_WRITE_ARBITRATION_MODE.request

The RCS_WRITE_ARBITRATION_MODE.request primitive shall permit an initiator to set the current arbitration mode for any routing device.

The parameters for this primitive shall be:

- RMAP_Parameters.
- Watchdog_TImeout

RMAP_Parameters includes the source address, destination address and transaction ID.

Arbitration_Mode shall contain the desired arbitration mode of the router (see Section 5.5.2.1.2).

5.5.5.8 RCS_WRITE_ARBITRATION_MODE.indication

The RCS_WRITE_ARBITRATION_MODE.indication primitive shall provide an indication to an initiator whether or not an arbitration mode value was written successfully in response to a RCS_WRITE_ARBITRATION_MODE.request.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

5.5.5.9 RCS_READ_TIME_COUNTER.request

The RCS_READ_TIME_COUNTER.request primitive shall permit an initiator to request the current time counter value for a specified channel on any routing device.

The parameters for this primitive shall be:

- RMAP_Parameters
- Time_Channel

RMAP_Parameters includes the source address, destination address and transaction ID.

Time_Channel shall specify the time counter channel (the value of the top two time-code bits) which should be read as part of this request (see Section 5.5.2.1.3).

5.5.5.10 RCS_READ_TIME_COUNTER.indication

The RCS_READ_TIME_COUNTER.indication primitive shall provide an indication to an initiator that a time counter value may be available in response to a RCS_READ_TIME_COUNTER.request. This primitive shall indicate whether the request was successful or not, and may contain a time counter value.

The parameters for this primitive shall be:

- Result
- Time_Counter
- Enabled

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Time_Counter parameter may be invalid.

Time_Counter shall contain the current time counter value of the channel specified during the RCS_READ_TIME_COUNTER.request primitive on this router (see Section 5.5.2.1.1).

Enabled shall contain the current enable flag value for the time counter associated with the channel specified during the RCS_READ_TIME_COUNTER.request primitive on this router (see Section 5.5.2.1.1).

5.5.5.11 RCS_RESET_TIME_COUNTER.request

The RCS_RESET_TIME_COUNTER.request primitive shall permit an initiator to request that the current time counter value for a specified channel on any routing device be reset to zero.

The parameters for this primitive shall be:

- RMAP_Parameters
- Time_Channel

RMAP_Parameters includes the source address, destination address and transaction ID.

Time_Channel shall specify the time counter channel (the value of the top two time-code bits) which should be reset as part of this request (see Section 5.5.2.1.3).

5.5.5.12 RCS_RESET_TIME_COUNTER.indication

The RCS_RESET_TIME_COUNTER.indication primitive shall provide an indication to an initiator that a time counter may have been reset in response to a RCS_RESET_TIME_COUNTER.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

5.5.5.13 RCS_ENABLE_TIME_COUNTER.request

The RCS_ENABLE_TIME_COUNTER.request primitive shall permit an initiator to write the current enable state for a specified channel on any routing device.

The parameters for this primitive shall be:

- RMAP_Parameters
- Time_Channel
- Enabled

RMAP_Parameters includes the source address, destination address and transaction ID.

Time_Channel shall specify the time counter channel (the value of the top two time-code bits) which should be reset as part of this request (see Section 5.5.2.1.3).

Enabled shall contain the desired enable flag value for the time counter associated with the channel specified by Time_Channel (see Section 5.5.2.1.1).

5.5.5.14 RCS_ENABLE_TIME_COUNTER.indication

The RCS_ENABLE_TIME_COUNTER.indication primitive shall provide an indication to an initiator that a time counter enable state may have been set in response to a RCS_ENABLE_TIME_COUNTER.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

5.5.5.15 RCS_READ_LA_COUNT.request

The RCS_READ_LA_COUNT.request primitive shall permit an initiator to request the count of supported logical addresses for any routing device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

5.5.5.16 RCS_READ_LA_COUNT.indication

The RCS_READ_LA_COUNT.indication primitive shall provide an indication to an initiator that a count of supported logical addresses may be available in response to a RCS_READ_LA_COUNT.request. This primitive shall indicate whether the request was successful or not, and may contain a logical address count.

The parameters for this primitive shall be:

- Result
- LA_Count

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the LA_Count parameter may be invalid.

LA_Count shall contain the number of logical addresses supported by the routing device (see Section 5.5.2.2.1).

5.5.5.17 RCS_READ_ROUTING_TABLE_ENTRY.request

The RCS_READ_ROUTING_TABLE_ENTRY.request primitive shall permit an initiator to request the current routing table information for any address on any routing device.

The parameters for this primitive shall be:

- RMAP_Parameters
- Address

RMAP_Parameters includes the source address, destination address and transaction ID.

Address shall be a physical or logical address value between 1 and 254 inclusively.

5.5.5.18 RCS_READ_ROUTING_TABLE_ENTRY.indication

The RCS_READ_ROUTING_TABLE_ENTRY.indication primitive shall provide an indication to an initiator that routing table information may be available in response to a RCS_READ_ROUTING_TABLE_ENTRY.request. This primitive shall indicate whether the request was successful or not, and may contain routing table information.

The parameters for this primitive shall be:

- Result
- Port_Association
- Address_Enabled
- Header_Deletion
- Group_Action
- Spill_If_Not_Ready
- Priority

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the LA_Count parameter may be invalid.

Port_Association shall contain a list of physical output ports associated with this address (see Section 5.5.2.2.2).

Address_Enabled shall indicate whether this address is enabled for routing (see Section 5.5.2.2.3).

Header_Deletion shall indicate if this address currently uses header deletion (see Section 5.5.2.2.3).

Group_Action shall indicate whether this address currently uses GAR or PD when routing to multiple output ports (see Section 5.5.2.2.3).

Spill_If_Not_Ready shall indicate if this address currently spills packets if output ports are not ready (see Section 5.5.2.2.3).

Priority shall indicate the current priority for this address as an integer (see Section 5.5.2.2.3).

5.5.5.19 RCS_WRITE_ROUTING_TABLE_ENTRY.request

The RCS_WRITE_ROUTING_TABLE_ENTRY.request primitive shall permit an initiator to write a routing table entry associated with any address on any device.

The parameters for this primitive shall be:

- RMAP_Parameters
- Address
- Port_Association
- Address_Enabled
- Header_Deletion
- Group_Action
- Spill_If_Not_Ready
- Priority

RMAP_Parameters includes the source address, destination address and transaction ID.

Address shall be a physical or logical address value between 1 and 254 inclusively.

Port_Association shall contain a list of physical output ports to be associated with this address (see Section 5.5.2.2.2).

Address_Enabled shall indicate whether this address should be enabled for routing (see Section 5.5.2.2.3).

Header_Deletion shall indicate if this address should use header deletion (see Section 5.5.2.2.3).

Group_Action shall indicate whether this address should use GAR or PD when routing to multiple output ports (see Section 5.5.2.2.3).

Spill_If_Not_Ready shall indicate if this address should spill packets if output ports are not ready (see Section 5.5.2.2.3).

Priority shall indicate the desired priority for this address as an integer (see Section 5.5.2.2.3).

5.5.5.20 RCS_WRITE_ROUTING_TABLE_ENTRY.indication

The RCS_WRITE_ROUTING_TABLE_ENTRY.indication primitive shall provide an indication to an initiator that a routing table entry may have been set in response to a RCS_WRITE_ROUTING_TABLE_ENTRY.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

5.5.6 Initiator Primitive Provision

Initiators shall provide primitives in accordance with the rules shown in Table 5-25. “M” indicates that provision of the primitive is mandatory; “O” indicates that provision of the primitive is optional. Where an optional request primitive is provided, an indication primitive shall also be provided, and vice-versa.

Table 5-25: Router Configuration Service Initiator Primitive Provision		
Primitive	**Provision**	**Notes**
RCS_READ_WATCHDOG_TIMEOUT.request	M	-
RCS_READ_WATCHDOG_TIMEOUT.indication	M	-
RCS_WRITE_WATCHDOG_TIMEOUT.request	M	-
RCS_WRITE_WATCHDOG_TIMEOUT.indication	M	-
RCS_READ_ARBITRATION_MODE.request	O	-
RCS_READ_ARBITRATION_MODE.request	O	-
RCS_WRITE_ARBITRATION_MODE.request	O	-
RCS_WRITE_ARBITRATION_MODE.request	O	-
RCS_READ_TIME_COUTER.request	M	-
RCS_READ_TIME_COUTER.indication	M	-
RCS_RESET_TIME_COUTER.request	O	-
RCS_RESET_TIME_COUTER.indication	O	-
RCS_ENABLE_TIME_COUNTER.request	M	-
RCS_ENABLE_TIME_COUNTER.indication	M	-
RCS_READ_LA_COUNT.request	M	-
RCS_READ_LA_COUNT.indication	M	-
RCS_READ_ROUTING_TABLE_ENTRY.request	M	-
RCS_READ_ROUTING_TABLE_ENTRY.indication	M	-
RCS_WRITE_ROUTING_TABLE_ENTRY.request	M	-
RCS_WRITE_ROUTING_TABLE_ENTRY.indication	M	-

5.6 TIME-CODE SOURCE SERVICE

5.6.1 Service Provision

Provision of the Time-Code Source service shall be optional for both targets and initiators.

5.6.2 Target Parameters

Time-Code Source service parameters are summarised in Table 5-26.

Table 5-26: Time-Code Source Service Parameters			
ID	**Name**	**Type**	**Summary**
0	Time-Code Sources	Complex	Control tick generation for up to 4 time-code sources
1-7	*Reserved*	-	*Reserved for future use*

5.6.2.1 Time-Code Sources

Time-Code Sources is a complex parameter permitting configuration of 1 to 4 independent time-code sources.

Fields in the root entry shall be as shown in Table 5-27.

Table 5-27: Time-Code Sources Parameter Root Entry Fields		
ID	**Name**	**Summary**
0	Time-Code Source Count	Count of time-code sources on this device
1-31	*Reserved*	*Reserved for future use*

Fields in non-root entries each represent a single time-code source and shall be as shown in Table 5-28. There shall be as many non-root entries as there are time-code sources on the device; if Time-Code Source service parameters are implemented, there shall be at least one (entry 1) and there shall be no more than 4. Valid entries shall be contiguous.

Table 5-28: Time-Code Sources Parameter Non-Root Entry Fields		
ID	**Name**	**Summary**
0	Generation Configuration	Configuration of time-code configuration
1	Generation Period	Time between generated time-codes
2-31	*Reserved*	*Reserved for future use*

5.6.2.1.1 Time-Code Source Count Field

The Time-Code Source Count field specifies the number of valid entries in the Time-Code Sources parameter. The count shall be represented as an unsigned number from 1 to 4, with the least

significant bit being bit 0. The field encoding is shown in Figure 5-20. Bits 2 to 31 are reserved for future use. The value of these bits shall be ignored. This field shall be read-only and constant.

Figure 5-20: Time-Code Source Count Field Encoding

5.6.2.1.2 Generation Configuration Field

The Generation Configuration field shall permit the time-code source associated with this entry to be configured. The Generation Configuration field shall be encoded as shown in Figure 5-21.

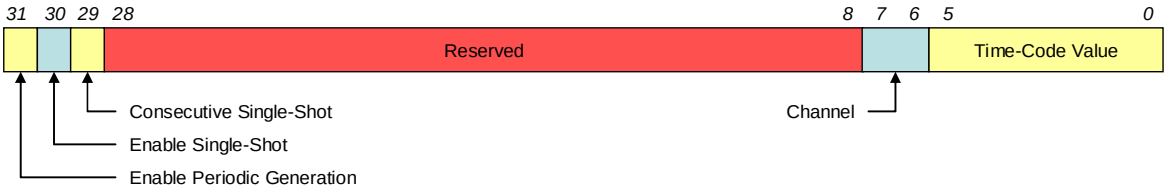


Figure 5-21: Generation Control Field Encoding

The Generation Control field shall have a number of parts:

- Bits 0 to 5 (where 0 is the least significant) shall contain the value that the next time-code to sent will contain. This shall default to zero, and shall increment automatically after each time-code sent.
- Bits 6 and 7 (where 6 is the least significant) shall contain the channel number (time-code flags value) for time-codes generated by this source.
- Bit 31 shall control the periodic generation of time-codes. When this bit is set to 1, time-codes shall be generated periodically, using the value of the Generation Period field. When this bit is set to 0, time-code shall not be generated periodically.
- Bit 30 shall always be read as 0; however, when this bit is written as a 1, a single time-code will be generated. The value of the time-code depends on the value of bit 29 in the written data.
- Bit 29 shall always read as 0; however, when this bit is written as a 1, and bit 30 is also written as a 1, the value of the time-code generated shall be that specified by the time-code value of this field (which shall then be incremented). If this bit is written as a 0, and bit 30 is written as a 1, the value of the time-code shall be specified by the value of bits 0 to 5 of the

written data. The time-code value in the field shall be updated to contain the value of the time-code sent, plus 1.

Bits 8 to 28 shall be reserved for future use. These bits shall be written as zero and shall be ignored when read.

5.6.2.1.3 Generation Period Field

The Generation Period field shall specify the period between time-codes when periodic generation is enabled. All 32 bits shall be valid and the field shall contain an unsigned number (where bit 0 is the least significant) indicating the time between consecutive time-codes in microseconds. Devices shall support a single contiguous range of periods; however, devices may choose not to support the full range. If a value higher than the maximum is written, the field shall be set to the maximum value; if a value lower than the minimum is written, the field shall be set to the minimum value.

5.6.3 Target Parameter Provision

When the Time-Code Source service is provided, device targets shall provide parameters in accordance with the rules shown in Table 5-29. "M" indicates that provision of the parameter/field for the identified operation is mandatory; "O" indicates that provision of the parameter/field for the identified operation is optional; "F" indicates that provision of the parameter/field for the identified operation is forbidden.

Table 5-29: Time-Code Source Service Target Parameter Provision

Parameter	Field(s)	Read	Write	RMW
Time-Code Sources	*Root entry fields*	M*	F	F
Time-Code Sources	*Non-root entry fields*	M	M	O

* All field values shall be constant.

5.6.4 Target Actions

A target shall not carry out any further actions beyond support for read/write/read-modify-write commands specified in 5.6.3

5.6.5 Initiator Primitives

A summary of Time-Code Source service initiator primitives is presented below:

- TCSS_READ_SOURCE_COUNT.request
- TCSS_READ_SOURCE_COUNT.indication

- TCSS_READ_SOURCE_CONFIG.request
- TCSS_READ_SOURCE_CONFIG.indication
- TCSS_WRITE_SOURCE_CONFIG.request
- TCSS_WRITE_SOURCE_CONFIG.indication
- TCSS_TC_SINGLE_SHOT.request
- TCSS_TC_SINGLE_SHOT.indication

These primitives are considered in more detail in the following sections.

5.6.5.1 TCSS_READ_SOURCE_COUNT.request

The TCSS_READ_SOURCE_COUNT.request primitive shall permit an initiator to request the count of supported time-code sources for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

5.6.5.2 TCSS_READ_SOURCE_COUNT.indication

The TCSS_READ_SOURCE_COUNT.indication primitive shall provide an indication to an initiator that a count of supported time-code sources may be available in response to a TCSS_READ_SOURCE_COUNT.request. This primitive shall indicate whether the request was successful or not, and may contain a time-code source count.

The parameters for this primitive shall be:

- Result
- Source_Count

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Source_Count parameter may be invalid.

Source_Count shall indicate the number of time-code sources supported by the device (see Section 5.6.2.1.1).

5.6.5.3 TCSS_READ_SOURCE_CONFIG.request

The TCSS_READ_SOURCE_CONFIG.request primitive shall permit an initiator to request configuration information for an identified time-code source for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters
- Source_Index

RMAP_Parameters includes the source address, destination address and transaction ID.

Source_Index shall contain the index of the source for which configuration information is being requested. Valid values for this parameter are integers between 1 and 4 inclusive.

5.6.5.4 TCSS_READ_SOURCE_CONFIG.indication

The TCSS_READ_SOURCE_CONFIG.indication primitive shall provide an indication to an initiator that configuration information of a single time-code source may be available in response to a TCSS_READ_SOURCE_CONFIG.request. This primitive shall indicate whether the request was successful or not, and may contain time-code source configuration information.

The parameters for this primitive shall be:

- Result
- Time_Code_Value
- Channel
- Enable_Periodic

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the other parameters may be invalid.

Time_Code_Value shall indicate the value of the next time-code that will be issued by this source, when required (see Section 5.6.2.1.2)

Channel shall indicate the current value that the top two bits of each time-code issued by this source shall take (see Section 5.6.2.1.2).

Enable_Periodic shall indicate whether or not periodic time-code generation is enabled for this source (see Section 5.6.2.1.2).

Period shall contain the current value of the periodic timer for this time-code source (see Section 5.6.2.1.3).

5.6.5.5 TCSS_WRITE_SOURCE_CONFIG.request

The TCSS_WRITE_SOURCE_CONFIG.request primitive shall permit an initiator to configure a time-code source on any device.

The parameters for this primitive shall be:

- RMAP_Parameters
- Source_Index
- Time_Code_Value
- Channel
- Enable_Periodic

RMAP_Parameters includes the source address, destination address and transaction ID.

Source_Index shall contain the index of the source for which configuration information is being written. Valid values for this parameter are integers between 1 and 4 inclusive.

Time_Code_Value shall indicate the value of the next time-code that should be issued by this source, when required (see Section 5.6.2.1.2)

Channel shall indicate the desired value that the top two bits of each time-code issued by this source shall take (see Section 5.6.2.1.2).

Enable_Periodic shall indicate whether or not periodic time-code generation should be enabled for this source (see Section 5.6.2.1.2).

Period shall contain the desired value of the periodic timer for this time-code source (see Section 5.6.2.1.3).

5.6.5.6 TCSS_WRITE_SOURCE_CONFIG.indication

The TCSS_WRITE_SOURCE_CONFIG.indication primitive shall provide an indication to an initiator that a time-code source may have been configured in response to a TCSS_WRITE_SOURCE_CONFIG.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

5.6.5.7 TCSS_TC_SINGLE_SHOT.request

The TCSS_TC_SINGLE_SHOT.request primitive shall permit an initiator to request that a time-code source generate a single time-code immediately. The primitive shall optionally permit the initiator to specify the value of the time-code.

The parameters for this primitive shall be:

- RMAP_Parameters
- Source_Index
- Time_Code_Value

RMAP_Parameters includes the source address, destination address and transaction ID.

Source_Index shall contain the index of the source which should be instructed to generate the time-code. Valid values for this parameter are integers between 1 and 4 inclusive.

Time_Code_Value shall indicate the value of the time-code that should be issued by this source (see Section 5.6.2.1.2). It shall be possible to specify a distinct value for this parameter such that the source automatically uses the next consecutive value for the time-code, rather than a value specified by this primitive.

5.6.5.8 TCSS_TC_SINGLE_SHOT.indication

The TCSS_TC_SINGLE_SHOT.indication primitive shall provide an indication to an initiator that a time-code source may have generated a time-code in response to a TCSS_TC_SINGLE_SHOT.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

5.6.6 Initiator Primitive Provision

When the Time-Code Source service is provided, initiators shall provide primitives in accordance with the rules shown in Table 7-11. “M” indicates that provision of the primitive is mandatory; “O” indicates that provision of the primitive is optional. Where an optional request primitive is provided, an indication primitive shall also be provided, and vice-versa.

Table 5-30: Router Configuration Service Initiator Primitive Provision		
Primitive	**Provision**	**Notes**
TCSS_READ_SOURCE_COUNT.request	M	-
TCSS_READ_SOURCE_COUNT.indication	M	-
TCSS_READ_SOURCE_CONFIG.request	M	-
TCSS_READ_SOURCE_CONFIG.indication	M	-
TCSS_WRITE_SOURCE_CONFIG.request	M	-
TCSS_WRITE_SOURCE_CONFIG.indication	M	-
TCSS_TC_SINGLE_SHOT.request	O	-
TCSS_TC_SINGLE_SHOT.indication	O	-

6 LEVEL 2 SERVICE DEFINITIONS

In general, devices supporting level 2 SpaceWire-PnP shall provide support for level 1 as defined in Section 5. Level 2 support makes additions to the parameters, fields, actions and primitives provided by the 5 core services defined in Section 5. No further services are added, and no restrictions are made on the use of services defined for level 1 support. Differences between level 1 and level 2 services are documented in this section.

6.1 DEVICE IDENTIFICATION SERVICE

6.1.1 Service Provision

Provision of the Device Identification service is mandatory for both targets and initiators.

6.1.2 Target Parameters

Device Identification parameters are summarised in Table 6-1.

Table 6-1: Level 2 Device Identification Service Parameters

ID	Name	Type	Differences to Level 1
0	Device Information	Simple	None
1	Vendor String	Simple	None
2	Product String	Simple	None
3	Device Status	Simple	Additional field to support device ownership
4	Capability List	Complex	None
5-7	*Reserved*	-	*Reserved for future use*

Parameters which differ from those specified for level 1 are described below.

6.1.2.1 Device Status

For level 2 devices, the Device Status is a parameter is comprised of the fields summarised in Table 6-2. Fields in the Device Status parameter shall be read-only.

Table 6-2: Level 2 Device Status Parameter Fields		
ID	**Name**	**Summary**
0	Network ID	Mirror of Network ID field
1	Active Ports	Mirror of Active Ports field
2	Return Port/Operational Status	Current return port number and device status
3	Device Owner	Mirror of Device Owner field
4-31	*Reserved*	*Reserved for future use*

The level 2 Device Status parameter shall have one additional field over and above those required for level 1 support. This field is described below.

6.1.2.1.1 Device Owner Field

The Device Owner field of the Device Status parameter shall contain a read-only mirror of the contents of the Device Owner field of the Network Discovery parameter provided by the Network Management service (see Section 6.2.2.1.1).

6.1.3 Target Parameter Provision

Level 2 device targets shall provide parameters in accordance with the rules defined for level 1 parameter provision (see Section 5.2.3).

6.1.4 Target Actions

A target shall not carry out any further actions beyond support for read/write/read-modify-write commands specified in Section 5.2.3

6.1.5 Initiator Primitives

Level 2 initiators shall provide all primitives defined for level 1 primitive provision; however, the following primitives are different for level 2 initiators.

- DIDS_READ_STATUS.indication

These primitives are considered in more detail in the following sections.

6.1.5.1 DIDS_READ_STATUS.indication

The DIDS_READ_STATUS.indication primitive shall provide an indication to an initiator that device status information may be available in response to a DIDS_READ_STATUS.request. This primitive shall indicate whether the request was successful or not, and may contain device status information.

The parameters for this primitive shall be:

- Result
- Network_ID
- Active_Ports
- Return_Port
- Operational_Status
- Owner_Data

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then all other parameters may be invalid.

Network_ID shall contain the current network ID of the device (see Section 5.2.2.4.1).

Active_Ports shall contain a list of currently active ports on the device (see Section 5.2.2.4.2).

Return_Port shall indicate the number of the port on the device which was used to respond for this indication (see Section 5.2.2.4.3).

Operational_Status shall contain the current operational status code for the device. Zero shall indicate that the device is fully operational; other codes are vendor specific for the device (see Section 5.2.2.4.3).

Device_Owner shall contain the current contents of the Device Owner field on the device (see Section 6.1.2.1.1).

6.1.6 Initiator Primitive Provision

Level 2 device initiators shall provide primitives in accordance with the rules defined for level 1 initiator primitive provision (see Section 5.2.6).

6.2 Network Management Service

6.2.1 Service Provision

Provision of the Network Management service is mandatory for both targets and initiators.

6.2.2 Target Parameters

Network Management parameters are summarised in Table 6-3.

Table 6-3: Level 2 Network Management Service Parameters			
ID	**Name**	**Type**	**Differences to Level 1**
0	Network Identification	Simple	Additional fields to support device ownership
1	Local Topology	Complex	Permits a level 2 proxy to describe the local topology
2-7	*Reserved*	-	*Reserved for future use*

Parameters which differ from, or are additional to, those specified for level 1 are described below.

6.2.2.1 Network Identification

Network Identification is a simple parameter comprised of the fields summarised in Table 6-4. Fields in the Network Identification parameter shall be read-write; except for the Region/Priority field, which shall be read-only.

Table 6-4: Level 2 Network Identification Parameter Fields		
ID	**Name**	**Summary**
0	Network ID	Permits an identifier to be assigned to this device
1	Device Logical Address	Logical address for this device
2	Device Owner	Device owner identifier, proxy ID and proxy key
3	Region/Priority	Owner region code and priority (proxy target only)
4-31	*Reserved*	*Reserved for future use*

The level 2 Network Identification parameter shall have two additional fields over and above those required for level 1 support; both are described below. Note that the Region/Priority field shall be provided by proxy targets only and shall not be provided by device targets.

6.2.2.1.1 Device Owner Field

The Device Owner field identifies the owner of this device (if any) and the proxy ID to be used for accessing the proxy provided by the owner. This field shall readable, using a read operation; and modifiable, using a read-modify-write operation only. Write operations shall not be supported. The

reset value of this field, indicating that the device is not owned, shall be zero. All bits shall be modifiable; however, reserved bits shall be written as zero and their value shall be ignored when read. Two formats for the field are shall be supported to facilitate networks using path or logical addressing.

Active nodes wishing to claim a device using logical addressing shall use the format shown in Figure 6-1. The Device Owner field shall comprise the following parts when logical addressing is used:

- Bit 31 shall always be zero for ownership using logical addressing.
- Bits 26 to 30 (where 26 is the least significant) shall provide the number of the port on the device that was used by the owner to access this device.
- Bits 16 to 23 (where 16 is the least significant) shall provide the logical address of the owner.
- Bits 8 to 15 (where 8 is the least significant) shall provide the proxy key to be used when accessing the owner-provided proxy from this device.
- Bits 0 to 7 (where 0 is the least significant) shall provide the proxy ID to be used when accessing the owner-provided proxy from this device.

Bits 24 and 25 shall be reserved for future use. These bits shall be written as zero and ignored when read.

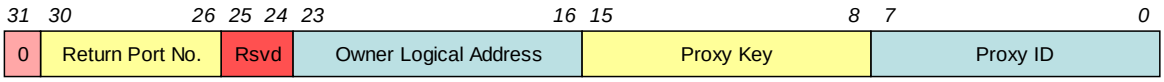


Figure 6-1: Device Owner Field Encoding for Logical Addressing

When the Device Owner field is to be modified, and the new value specifies the logical addressing format, routing devices shall update the routing table to provide an entry which maps the Owner Logical Address onto the Return Port Number. This is specified by the Device Owner Address target action (see Section 6.2.4.1). If the state of the link specified by the Return Port Number changes from active to inactive (i.e. the link disconnects), the device reset the value of the Device Owner field to zero. This is specified by the Owner Link Disconnect target action (see Section 6.2.4.2).

Active nodes wishing to claim a device using path addressing shall use the format shown in Figure 6-2. The Device Owner field shall comprise the following parts when logical addressing is used:

- Bit 31 shall always be one for ownership using path addressing.
- Bits 26 to 30 (where 26 is the least significant) shall provide the number of the port on the device that was used by the owner to access this device. The forms the first part of the path address to the owner.

- Bits 21 to 25 (where 21 is the least significant) shall provide the second part of the path address (or zero, if no further addresses are necessary).
- Bits 16 to 20 (where 16 is the least significant) shall provide the third part of the path address (or zero, if no further addresses are necessary).
- Bits 8 to 15 (where 8 is the least significant) shall provide the proxy key to be used when accessing the owner-provided proxy from this device.
- Bits 0 to 7 (where 0 is the least significant) shall provide the proxy ID to be used when accessing the owner-provided proxy from this device.

Bits 21 and 22 are reserved for future use. These bits shall be written as zero and ignored when read.

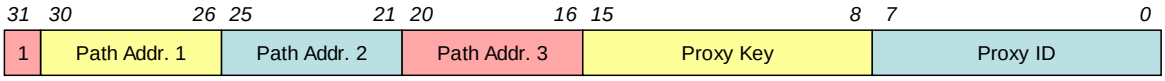


Figure 6-2: Device Owner Field Encoding for Path Addressing

If the state of the link specified by Path Address 1 changes from active to inactive (i.e. the link disconnects), the device reset the value of the Device Owner field to zero. This is specified by the Owner Link Disconnect target action (see Section 6.2.4.2).

A read-modify-write operation on this field via the Owner Proxy service shall be used to indicate that ownership has been claimed by another node, providing the new contents of the Device Owner field.

6.2.2.1.2 Region/Priority Field

The Region/Priority field shall provide the current region for this device and the owner priority for ownership arbitration. This field shall only be provided by proxy targets; devices shall not provide this field. The Region/Priority field shall be encoded as shown in Figure 6-3.

Figure 6-3: Region/Priority Field Encoding

The Region/Priority field shall have the following parts:

- Bits 31 to 8 (where 8 is the least significant) shall identify the region that this device has been included in (usually the region of the owner) and shall be non-zero.

- Bits 7 to 0 (where 0 is the least significant) shall identify the priority of the owner of this device.

The Region/Priority field shall by read-only.

6.2.2.2 Local Topology

Local Topology is an optional complex parameter provided by proxy targets only and supplying information about the topology local to the device being proxied. All fields in both root and non-root entries in this parameter are read-only.

Fields in the root entry shall be as shown in Table 6-5.

Table 6-5: Local Topology Parameter Root Entry Fields

ID	Name	Summary
0	Device Count	Count of devices in this topology table
1-31	*Reserved*	*Reserved for future use*

Fields in non-root entries each represent a single device and shall be as shown in Table 6-6. There shall be as many non-root entries as there are devices in this region, or as much of the network as the owner device is aware of, up to the maximum of 254 devices. Entry 1 shall represent the device being proxied; entry 255 is reserved and shall not be used. Valid entries shall be contiguous.

Table 6-6: Local Topology Parameter Non-Root Entry Fields		
ID	**Name**	**Summary**
0	Vendor ID/ Product ID	Mirror of Vendor ID/Product ID
1	Region/Number of Ports	Mirror of Region/Number of Ports
2	Device ID High	Mirror of Device ID High
3	Device ID Low	Mirror of Device ID Low
4	Version/Instance ID	Mirror of Version/Instance ID
5	Network ID	Mirror of Network ID field
6	Device Owner	Mirror of Device Owner (level 2 devices only)
7	*Reserved*	*Reserved for future use*
8	Connection Data (Ports 0 and 1)	Local Topology entry references for port connections
9	Connection Data (Ports 2 and 3)	Local Topology entry references for port connections
10	Connection Data (Ports 4 and 5)	Local Topology entry references for port connections
11	Connection Data (Ports 6 and 7)	Local Topology entry references for port connections
12	Connection Data (Ports 8 and 9)	Local Topology entry references for port connections
13	Connection Data (Ports 10 and 11)	Local Topology entry references for port connections
14	Connection Data (Ports 12 and 13)	Local Topology entry references for port connections
15	Connection Data (Ports 14 and 15)	Local Topology entry references for port connections
16	Connection Data (Ports 16 and 17)	Local Topology entry references for port connections
17	Connection Data (Ports 18 and 19)	Local Topology entry references for port connections
18	Connection Data (Ports 20 and 21)	Local Topology entry references for port connections
19	Connection Data (Ports 22 and 23)	Local Topology entry references for port connections
20	Connection Data (Ports 24 and 25)	Local Topology entry references for port connections
21	Connection Data (Ports 26 and 27)	Local Topology entry references for port connections
22	Connection Data (Ports 28 and 29)	Local Topology entry references for port connections
23	Connection Data (Ports 30 and 31)	Local Topology entry references for port connections

24-31	*Reserved*	*Reserved for future use*

6.2.2.2.1 Device Count Field

The Device Count field specifies the number of valid entries in the Local Topology parameter. The count shall be represented as an unsigned number from 1 to 254, with the least significant bit being bit 0. The field encoding is shown in Figure 6-4. Bits 8 to 31 are reserved for future use. The value of these bits shall be ignored.

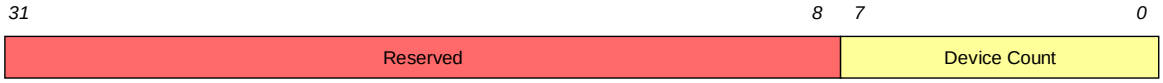


Figure 6-4: Device Count Field Encoding

6.2.2.2.2 Device Information Mirror Fields

The first 5 fields of each non-root entry of the Local Topology parameter shall provide a mirror of the first 5 fields of the Device Information parameter (Device Identification service) for the device that this entry represents.

6.2.2.2.3 Device Status Mirror Fields

Field IDs 5 and 6 shall provide a mirror of the Network ID and Device Owner fields respectively of the Device Status parameter (Device Identification service) for the device that this entry represents. Where this entry represents a level 1 device, the Device Owner field (field ID 6) shall be reserved.

6.2.2.2.4 Connection Data Fields

The Connection Data fields shall specify the entry number of the device to which the relevant port on the device this entry represents is connected to. Each entry provides the connection for two physical ports. In addition to the entry number for the connected device, the field shall specify the port on the remote device that is used for the connection. Where a port is not connected, or it is the configuration port (port zero), the connection entry and remote port shall both be specified as 0. Where a port is connected to a device not listed in this Local Topology parameter (either due to lack of space, or due to incomplete network discovery), the connection entry shall be specified as 255 and the remote port shall be specified as zero. Connection Data fields shall be encoded as shown in Figure 6-5.

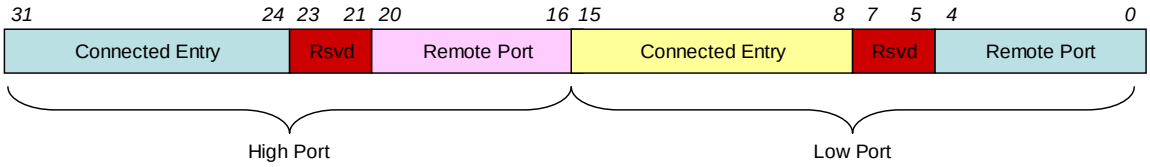


Figure 6-5: Connection Data Field Encoding

Connection Data fields shall each represent two connections, each connection requiring 16 bits. The higher numbered connection shall be represented by bits 16 to 31, the lower numbered connection shall be represented by bits 0 to 15.

- Bits 24 to 31 (where 24 is the least significant) shall contain the entry number of the device to which the higher numbered port is connected. If the port is not connected, this shall be zero. If the port is connected to a device not represented by this Local Topology parameter, this shall be 255.
- Bits 16 to 20 (where 16 is the least significant) shall contain the physical port number on the remote device which is used for the higher numbered connection. This port number must be a valid SpaceWire or external port for the remote device. If the Connected Entry is either zero or 255, this field shall be zero.
- Bits 8 to 15 (where 8 is the least significant) shall contain the entry number of the device to which the lower numbered port is connected. If the port is not connected, this shall be zero. If the port is connected to a device not represented by this Local Topology parameter, this shall be 255.
- Bits 0 to 4 (where 0 is the least significant) shall contain the physical port number on the remote device which is used for the lower numbered connection. This port number must be a valid SpaceWire or external port for the remote device. If the Connected Entry is either zero or 255, this field shall be zero.

Bits 5 to 7 and 21 to 23 are reserved. The value of these bits shall be ignored and shall always be written as zero. These bits are reserved for future use.

For example, if port 2 is connected to port 4 of the device with entry index 6, and port 3 is not connected, the field with ID 7 would contain 0x00000604. Similarly, if port 10 is connected to a device not included in this Local Topology parameter, and if port 11 is connected to port 17 of the device with entry index 160, the field with ID 11 would contain 0xA011FF00.

6.2.3 Target Parameter Provision

Level 2 device targets shall provide parameters in accordance with the rules defined for level 1 parameter provision (see Section 5.3.3) with additional level 2 fields and parameters provided in accordance with the rules shown in Table 6-7. "M" indicates that provision of the parameter/field for the identified operation is mandatory; "O" indicates that provision of the parameter/field for the identified operation is optional; "F" indicates that provision of the parameter/field for the identified operation is forbidden; "PM" indicates that provision of the parameter/field for the identified operation is mandatory for proxy targets and forbidden for device targets; "PM" indicates that provision of the parameter/field for the identified operation is optional for proxy targets and forbidden for device targets.

Table 6-7: Network Management Service Level 2 Target Parameter Provision

Parameter	Field(s)	Read	Write	RMW
Network Identification	Device Owner	M	F	M
Network Identification	Region/Priority	PM	F	F
Local Topology	*Root entry fields*	PM	F	F
Local Topology	*Non-root entry fields*	PO	F	F

6.2.4 Target Actions

In addition to support for target actions specified for level 1 support (see Section 5.3.4), level 2 targets shall support read/write/read-modify-write commands specified in Section 6.2.3. Additionally, routing devices shall support the Device Owner Address action defined below.

6.2.4.1 Device Owner Address

The Device Owner Address action applies only to routing devices and shall be initiated whenever an attempt is made to modify the Device Owner field of the Network Identification parameter, and the new value specifies the logical addressing format. The action shall cause the routing table to be updated to provide an entry which maps the Owner Logical Address onto the Return Port Number. This shall replace any previous mapping associated with that logical address. No other routing table entry shall be affected. If the router does not support mapping for the specified logical address, the read-modify-write operation shall return the error "RMAP Command not implemented or not authorised" in reply to the modification operation and the contents of the Device Owner field shall not be affected.

6.2.4.2 Owner Link Disconnect

The Owner Link Disconnect action applies to all devices and shall be initiated if the state of the link identified by the Return Port Number/Path Address 1 part of the Device Owner field of the Network Identification parameter changes from active to inactive (i.e. the link disconnects). The action shall cause the Device Owner field of the Network Identification parameter to be reset to zero. A change in the state of any other link shall not affect the value of this field.

6.2.5 Initiator Primitives

Level 2 initiators shall provide all primitives defined for level 1 primitive provision; however, the following primitives are different for level 2 initiators.

- NMS_DISCOVER_NETWORK.request
- NMS_DISCOVER_NETWORK.indication

The following services are defined in addition to those defined for level 1 support:

- NMS_READ_DEVICE_OWNER.request
- NMS_READ_DEVICE_OWNER.indication
- NMS_MODIFY_DEVICE_OWNER.request
- NMS_MODIFY_DEVICE_OWNER.indication
- NMS_READ_REGION_PRIORITY.request
- NMS_READ_REGION_PRIORITY.indication
- NMS_READ_LOCAL_TOPOLOGY.request
- NMS_READ_LOCAL_TOPOLOGY.indication
- NMS_OWNER_IDENTIFICATION.request
- NMS_OWNER_IDENTIFICATION.indication
- NMS_CLAIM_DEVICE.request
- NMS_CLAIM_DEVICE.indication
- NMS_RELEASE_DEVICE.request
- NMS_RELEASE_DEVICE.indication
- NMS_FORCE_RELEASE.indication
- NMS_GENERATE_PROXY.request

- NMS_GENERATE_PROXY.indication
- NMS_RELEASE_PROXY.request
- NMS_RELEASE_PROXY.indication
- NMS_PROXY_REQUEST.indication
- NMS_PROXY_REPLY.request
- NMS_PROXY_REPLY.indication

These primitives are considered in more detail in the following sections.

6.2.5.1 NMS_DISCOVER_NETWORK.request

The NMS_DISCOVER_NETWORK.request primitive shall initiate a breadth-first traversal of the network, gathering basic device and topology information which shall be supplied to the initiator as part of the NMS_DISCOVER_NETWORK.indication primitive.

The parameters for this primitive shall be:

- Size_Limit

Size_Limit specifies a limit (as a number of devices) for the network traversal. It shall be possible to specify a value for this parameter such that no limit is placed on the network traversal.

The algorithm used by this primitive shall be based on a breath-first traversal of the network using the following mechanism to identify devices already discovered (indicating loops or alternative routes in the network topology):

1. If the Device Information parameter of the Device Identification service provides a Device ID then this shall be used to uniquely identify this device.
2. If a Device ID is not provided, the Device Owner field in the Device Status parameter shall be read using the Device Identification service. If this field is non-zero, the validity of the current owner should be verified using the algorithm specified for the NMS_OWNER_IDENTIFICATION.request primitive. If the owner exists then the combination of the Device Owner field and the region code provided by the owner shall be used to uniquely identify this device.
3. If a Device ID is not provided, and the device has no valid owner, the network discovery algorithm shall not continue. Note that taking ownership of the device would provide it with a valid owner, permitting the algorithm to continue.

6.2.5.2 NMS_DISCOVER_NETWORK.indication

The NMS_DISCOVER_NETWORK.indication primitive shall indicate to an initiator that a network topology may be available in response to a NMS_DISCOVER_NETWORK.request. This primitive shall indicate whether the request was successful or not, and may contain a device connectivity list identifying network devices and their connectivity and, by extension, the network topology.

The parameters for this primitive shall be:

- Result
- Device_Connectivity

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Device_Connectivity parameter may be invalid.

Device_Connectivity shall contain an enumerated list of devices discovered on the network, together with their connectivity. Each entry in the list describes a single network device and shall contain the following parameters:

- Vendor_ID
- Product_ID
- Preferred_Region
- Router_Node
- Support_Level
- Port_Count
- Device_ID
- Version
- Instance_ID
- Network_ID
- Device_Owner
- Link_List

Vendor_ID shall contain the vendor ID of the device (see Section 5.2.2.1.1).

Product_ID shall contain the product ID of the device (see Section 5.2.2.1.1).

Preferred_Region shall contain the preferred region identifier for the device; this may be zero if no region is preferred (see Section 5.2.2.1.2).

Router_Node shall indicate whether the device is a router or a node (see Section 5.2.2.1.2).

Support_Level shall indicate whether this device supports SpaceWire-PnP level 1 or 2 (see Section 5.2.2.1.2).

Port_Count shall contain the number of physical ports on the device (see Section 5.2.2.1.2).

Device_ID shall contain the device ID; this may be zero if no device ID is provided (see Section 5.2.2.1.3).

Version shall contain the version information for the device (see Section 5.2.2.1.4).

Instance_ID shall contain the instance ID for the device (see Section 5.2.2.1.4).

Network_ID shall contain the current network ID of the device (see Section 5.3.2.1.1).

Device_Owner shall contain the current device owner information for the device (see Section 6.2.2.1.1) when the device is a level 2 device. If the device is a level 1 device, this parameter shall be invalid.

Link_List shall contain an enumerated list, where each entry represents a physical port on this device. There shall be an entry for every physical port. Each entry shall have the following parameters:

- Connected_Device
- Connected_Port

Connected_Device shall reference an entry in the Device_Connectivity list indicating the device to which the physical port represented by this entry is connected. If the physical port represented by this entry is not connected, this parameter shall hold a distinct value (such as zero) to indicate this. Additionally, if the physical port represented by this entry is connected to a device not included in the Device_Connectivity list, this parameter shall hold a distinct value (such as -1) to indicate this.

Connected_Port shall indicate the physical port number on the device to which the physical port represented by this entry is connected (i.e. the physical port number on the remote device). If The Connected_Device parameter indicates that the physical port represented by this entry is either not connected, or is connected to a device not included in the Device_Connectivity list, this parameter may not be valid.

6.2.5.3 NMS_READ_DEVICE_OWNER.request

The NMS_READ_DEVICE_OWNER.request primitive shall permit an initiator to request the current contents of the Device Owner field for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

6.2.5.4 NMS_READ_DEVICE_OWNER.indication

The NMS_READ_DEVICE_OWNER.indication primitive shall provide an indication to an initiator that Device Owner field data may be available in response to a NMS_READ_DEVICE_OWNER.request. This primitive shall indicate whether the request was successful or not, and may contain device owner data.

The parameters for this primitive shall be:

- Result
- Device_Owner

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Device_Owner parameter may be invalid.

Device_Owner shall contain the current device owner field data for the device (see Section 6.2.2.1.1).

6.2.5.5 NMS_MODIFY_DEVICE_OWNER.request

The NMS_MODIFY_DEVICE_OWNER.request primitive shall permit an initiator to attempt to modify the current contents of the Device Owner field for any network device.

The parameters for this primitive shall be:

- RMAP_Parameters;
- Current_Device_Owner;
- New_Device_Owner.

RMAP_Parameters includes the source address, destination address and transaction ID.

Current_Device_Owner shall contain the current device owner field data for the device (see Section 6.2.2.1.1). This is used as part of the read-modify-write operation in order to prevent race conditions during ownership competition.

New_Device_Owner shall contain the desired device owner field data for the device (see Section 6.2.2.1.1). This will be written to the device only if the contents of the Current_Device_Owner parameter matches the contents of the Device_Owner field.

6.2.5.6 NMS_MODIFY_DEVICE_OWNER.indication

The NMS_MODIFY_DEVICE_OWNER.indication primitive shall provide an indication to an initiator that the Device Owner field data may have been modified in response to a NMS_MODIFY_DEVICE_OWNER.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result.

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

6.2.5.7 NMS_READ_REGION_PRIORITY.request

The NMS_READ_REGION_PRIORITY.request primitive shall permit an initiator to request the current region and owner priority for any network device by interrogating the device proxy.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

6.2.5.8 NMS_READ_REGION_PRIORITY.indication

The NMS_READ_REGION_PRIORITY.indication primitive shall provide an indication to an initiator that current region and owner priority information may be available in response to a NMS_READ_REGION_PRIORITY.request. This primitive shall indicate whether the request was successful or not, and may contain region and priority information.

The parameters for this primitive shall be:

- Result
- Region
- Priority

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Region_Priority parameter may be invalid.

Region shall contain the current region for the device (see Section 6.2.2.1.2) obtained from the owner proxy.

Priority shall contain the priority of the owner of the device (see Section 6.2.2.1.2) obtained from the owner proxy.

6.2.5.9 NMS_READ_LOCAL_TOPOLOGY.request

The NMS_READ_LOCAL_TOPOLOGY.request primitive shall permit an initiator to request local topology information for any network device (if available) by interrogating the device proxy.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

6.2.5.10 NMS_READ_LOCAL_TOPOLOGY.indication

The NMS_READ_LOCAL_TOPOLOGY.indication primitive shall provide an indication to an initiator that local topology information may be available in response to a NMS_READ_LOCAL_TOPOLOGY.request. This primitive shall indicate whether the request was successful or not, and may contain local topology information.

The parameters for this primitive shall be:

- Result
- Local_Topology

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Region_Priority parameter may be invalid.

Local_Topology shall contain an enumerated list of network devices and connectivity, extracted from local topology information provided by the owner proxy for the device specified as part of the NMS_READ_LOCAL_TOPOLOGY.request primitive. Each entry in the list describes a single network device and shall contain the following parameters:

- Vendor_ID
- Product_ID
- Preferred_Region
- Router_Node
- Support_Level

- Port_Count
- Device_ID
- Version
- Instance_ID
- Network_ID
- Link_List

Vendor_ID shall contain the vendor ID of the device (see Section 5.2.2.1.1).

Product_ID shall contain the product ID of the device (see Section 5.2.2.1.1).

Preferred_Region shall contain the preferred region identifier for the device; this may be zero if no region is preferred (see Section 5.2.2.1.2).

Router_Node shall indicate whether the device is a router or a node (see Section 5.2.2.1.2).

Support_Level shall indicate whether this device supports SpaceWire-PnP level 1 or 2 (see Section 5.2.2.1.2).

Port_Count shall contain the number of physical ports on the device (see Section 5.2.2.1.2).

Device_ID shall contain the device ID; this may be zero if no device ID is provided (see Section 5.2.2.1.3).

Version shall contain the version information for the device (see Section 5.2.2.1.4).

Instance_ID shall contain the instance ID for the device (see Section 5.2.2.1.4).

Network_ID shall contain the current network ID of the device (see Section 5.3.2.1.1).

Link_List shall contain an enumerated list, where each entry represents a physical port on this device. There shall be an entry for every physical port. Each entry shall have the following parameters:

- Connected_Device
- Connected_Port

Connected_Device shall reference an entry in the Local_Topology list indicating the device to which the physical port represented by this entry is connected. If the physical port represented by this entry is not connected, this parameter shall hold a distinct value (such as zero) to indicate this. Additionally, if the physical port represented by this entry is connected to a device not included in the Local_Topology list, this parameter shall hold a distinct value (such as -1) to indicate this.

Connected_Port shall indicate the physical port number on the device to which the physical port represented by this entry is connected (i.e. the physical port number on the remote device). If The Connected_Device parameter indicates that the physical port represented by this entry is either not connected, or is connected to a device not included in the Device_Connectivity list, this parameter may not be valid.

6.2.5.11 NMS_OWNER_IDENTIFICATION.request

The NMS_OWNER_IDENTIFICATION.request primitive shall permit an initiator to request the full SpaceWire address from any network device to its owner, as well as the device region and owner priority, ensuring that the identified owner is valid.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

The following algorithm shall be used:

1. The Device Owner field of the Device Status parameter shall be read using the Device Identification service. This shall be done directly (not to the owner proxy). If no response is received from the device within timeout T1, it shall be assumed that the device is no longer present on the network, or is not functioning correctly.
2. If the owner address specified by the Device Owner field is a logical address, the address to the owner shall be composed of the address to the device (if it is a router), or to the router connected to the return port specified by the Device Owner field (in the case of a node), followed by the logical address specified by the Device Owner field.
3. If the owner address is a path address, the complete path may not have been specified by the Device Owner field as it contains a maximum of three path addresses. If the path address specified by the Device Owner field contains three port numbers (three 'hops'), the path may not be complete. The Device Identification service shall be used to read the Number of Ports/Region parameter of the device specified by this path which indicates whether the path addresses a node or a router. If a node is addressed by the path, the address is complete, if a router is addressed, the path from the router to the owner shall be determined, again by reading the Device Owner field of the Device Status parameter using the Device Identification service (in a similar manner to. step 1). The path from the original device and the path specified by the router shall be combined to form a complete path to the router. This process shall be repeated until the complete path to the owner has been constructed.

4. The owner address constructed in steps 2 and 3 shall be used to read the Region/Priority field of the Network Identification parameter provided by the Network Management service on the proxy target.
5. If the device is not a level 1 router, and no response is received from the owner within timeout T1, it shall be assumed that the owner is no longer present on the network, or is not functioning correctly. If the device is a level 1 router and no response is received from the owner within timeout T1 then the read of the Region/Priority field on the proxy target (step 4) shall be retried at least once.

The resulting validity information, addressing information, region and priority shall be supplied to the initiator using a NMS_OWNER_IDENTIFICATION.indication primitive.

Timeouts values are described in Section 6.2.7

6.2.5.12 NMS_OWNER_IDENTIFICATION.indication

The NMS_OWNER_IDENTIFICATION.indication primitive shall provide an indication to an initiator that validity, address, region and priority information may be available in response to a NMS_OWNER_IDENTIFICATION.request. This primitive shall indicate whether the request was successful or not, and may contain owner identification information.

The parameters for this primitive shall be:

- Result;
- Owner_Valid
- Port;
- Address;
- Region;
- Priority.

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then all other parameters may be invalid.

Owner_Valid shall indicate whether the owner is a valid node on the network. If Owner_Valid indicates that the owner is not valid, the Port, Address, Region and Priority parameters will be invalid and shall be ignored.

Port shall contain the physical port number which is being used by the current owner to access this device.

Address shall contain the full SpaceWire address for the owner of the device.

Region shall contain the current region of the device, obtained from the owner proxy.

Priority shall contain the priority of the owner.

6.2.5.13 NMS_CLAIM_DEVICE.request

The NMS_CLAIM_DEVICE.request primitive shall permit an initiator to request ownership of any network device.

The parameters for this primitive shall be:

- RMAP_Parameters;
- Region;
- Priority;
- Proxy_ID;
- Proxy_Key.

RMAP_Parameters includes the source address, destination address and transaction ID.

Region specifies the preferred region of the initiator.

Priority specifies the priority of the initiator.

Proxy_ID specifies the proxy ID to be assigned to the device if it is successfully claimed.

Proxy_Key specifies the proxy key to be assigned to the device if it is successfully claimed.

The exact algorithm to be used depends on the support level of the device. In the simplest case, where the device is a level 2 device, the following algorithm shall be used for claiming the device:

1. The owner of the device (with region and priority information) shall be identified using the same algorithm as that used for the NMS_OWNER_IDENTIFICATION.request primitive (see Section 6.2.5.11).
2. If the device has a current owner, competition for the device shall be resolved. This utilises the preferred region of the device (from the Number of Ports/Region field of the Device Information parameter provided by the Device Identification service); the region of the remote node, the priority of the current owner and the Region and Priority parameters supplied to this primitive. It may also utilise the physical port number which is being used for access by the current owner and the initiator (from the Return Port/Operational Status field of the Device Status parameter provided by the Device Identification service). The competition resolution algorithm shall be as follows:

a. The device region shall be compared to that of the two potential owners:
    i. If the region of one node matches that of the device, ownership shall be granted to that node.
    ii. If the region of neither node matches that of the device, or if the region of both nodes match that of the device, then ownership shall be determined by priority.
b. If necessary, the priorities of the two potential owners shall be compared:
    i. If the priority of one node is greater than that of the other, ownership shall be granted to that node.
    ii. If the priorities of the two nodes are equal then ownership shall be determined by return port number.
c. If necessary, the return port numbers of the two potential owners shall be compared. The node with the numerically lowest return port number shall be granted ownership of the device.

3. If the device has no current owner, or if competition resolution found in favour of the initiator, the device may be claimed. This shall be accomplished as follows:
    a. The proxy information (proxy ID and proxy key), along with the address to the initiator from the device, shall be written to the Device Owner field of the Network Identification parameter using a read-modify-write operation. This shall be done directly to the device (not to the proxy target).
    b. If the read-modify-write operation fails there is either competition for the device, or it is no longer available, in both cases failure of the read-modify-write operation shall cause the device claim algorithm to be restarted (from step 1).
    c. If the read-modify-write operation succeeds then the device has been claimed.
    d. If the device did have a previous, valid, owner, the new Device Owner field value shall be written to Device Owner field of the Network Identification parameter on the proxy target provided by the previous owner of the device. This informs the previous owner that it has lost ownership.

Devices shall only be claimed contiguously and providing they have been fully discovered. Specifically:

- A node shall not make an attempt to claim a device unless there is at least one path between the node and the device composed entirely of routers owned by the node. If the device is directly connected to the node an ownership attempt shall be permitted.
- A node shall not make an attempt to claim a device unless is has fully discovered all other devices directly connected to the device in question.

The success, or otherwise, of this primitive shall be indicated to the initiator using an NMS_CLAIM_DEVICE.indication primitive.

If the device is a level 1 device, the read-modify-write command may not be supported, and a the device will not support the Device Owner Address and Owner Link Disconnect target actions. These limitations result in the need for a number of additional timeouts to protect against race conditions. When interrogating level 1 devices, the algorithm above shall be used where the read-modify-write operation in step 4 shall be replaced with the following algorithm:

a. The Device Owner field shall be read.

b. If the result of the read operation matches the owner details determined during step 1, the new Device Owner field shall be written within timeout T2. If the result of the read operation does not match the device claim algorithm shall be restarted.

c. Before issuing a read command to verify the correct operation of the write command, a delay specified by timeout T3 shall be inserted.

d. The Device Owner field shall be read to verify that the new owner details have been written correctly.

e. If the result of the read command does not match the owner details written then the device claim algorithm shall be restarted. If the result of the read command does match the new owner details then the device has been claimed.

As in step 4.d. in the algorithm for level 2 devices, when a level 1 device is claimed, a previous owner must be informed. Once a level 1 router has been successfully claimed, if logical addressing has been specified for the owner details then a write command to configure the necessary routing table entry shall be issued within timeout T2.

Timeouts values are described in Section 6.2.7

6.2.5.14 NMS_CLAIM_DEVICE.indication

The NMS_CLAIM_DEVICE.indication primitive shall provide an indication to an initiator that success information may be available in response to a NMS_CLAIM_DEVICE.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result.

Result shall indicate the success, or otherwise, of the NMS_CLAIM_DEVICE.request.

6.2.5.15 NMS_RELEASE_DEVICE.request

The NMS_RELEASE_DEVICE.request primitive shall permit an initiator to release ownership of any device.

The parameters for this primitive shall be:

- RMAP_Parameters.

RMAP_Parameters includes the source address, destination address and transaction ID.

To release ownership of the device, the Device Owner field of the Network Identification parameter shall be set to zero.

Note that:

- This operation shall not be carried out on devices that the initiator does not own.
- If the initiator wishes to release ownership of a router that would prevent contiguous ownership of other devices, the node shall release ownership of all relevant devices in an order that ensures device ownership is contiguous at all times.

6.2.5.16 NMS_RELEASE_DEVICE.indication

The NMS_RELEASE_DEVICE.indication primitive shall provide an indication to an initiator that success information may be available in response to a NMS_RELEASE_DEVICE.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result.

Result shall indicate the success, or otherwise, of the NMS_RELEASE_DEVICE.request.

6.2.5.17 NMS_FORCE_RELEASE.indication

The NMS_FORCE_RELEASE.indication primitive shall indicate to an initiator that the claim of ownership on a device has been forcibly released due a loss of ownership to a competing device.

The parameters for this primitive shall be:

- Device_Owner.

Device_Owner shall contain the new device owner field data for the device (see Section 6.2.2.1.1).

6.2.5.18 NMS_GENERATE_PROXY.request

The NMS_GENERATE_PROXY.request primitive shall permit an initiator to request the generation of a valid proxy target to support device ownership.

This primitive shall the following parameters:

- Proxy_ID;
- Proxy_Key.

Proxy_ID shall specify the desired proxy ID for the new proxy target. This may be zero, in which case a proxy ID shall be generated randomly.

Proxy_Key shall specify the desired proxy key for the new proxy target. This may be zero, in which case a proxy key shall be generated randomly.

An initiator may support up to 255 proxies which may each be uniquely identified by the proxy ID. If an initiator is able to guarantee (by design) that a given proxy ID/key combination is network unique then values may be specified as part of this request primitive. If an initiator is unable to make this guarantee, both values shall be set to zero, and this primitive shall generate random values.

6.2.5.19 NMS_GENERATE_PROXY.indication

The NMS_GENERATE_PROXY.indication primitive shall indicate the successful generation, or otherwise, of a proxy target in response to a NMS_GENERATE_PROXY.request primitive. This primitive shall indicate whether the request was successful or not, and may contain a proxy ID and proxy key.

This primitive shall have the following parameters:

- Result;
- Proxy_ID;
- Proxy_Key.

Result shall indicate the success, or otherwise, of this primitive. If Result does not indicate success then the Proxy_ID and Proxy_Key parameters shall be invalid.

Proxy_ID specifies the proxy ID assigned to the newly generated proxy target.

Proxy_Key specifies the proxy key assigned to the newly generated proxy target.

6.2.5.20 NMS_RELEASE_PROXY.request

The NMS_RELEASE_PROXY.request primitive shall permit an initiator to request the release of resources associated with a proxy previously generated using the NMS_GENERATE_PROXY.request primitive.

The parameters for this primitive shall be:

- Proxy_ID.

Proxy_ID specifies the ID of the proxy for which to release resources.

6.2.5.21 NMS_RELEASE_PROXY.indication

The NMS_RELEASE_PROXY.indication primitive shall provide an indication to an initiator that the resources associated with a proxy may have been successfully released in response to a NMS_RELEASE_PROXY.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result.

Result shall indicate the success, or otherwise, of this primitive.

6.2.5.22 NMS_PROXY_REQUEST.indication

The NMS_PROXY_REQUEST.indication primitive shall indicate to an initiator that a command has been received directed to a proxy target.

The parameters for this primitive shall be:

- RMAP_Parameters;
- Operation;
- Proxy_ID;
- Proxy_Key;
- Service_ID;
- Parameter_ID;
- Entry;
- Field_ID;
- Length;

- Data;
- Mask.

RMAP_Parameters includes the source address, destination address and transaction ID.

Operation specifies the type of operation requested and shall be one of the following: read; verified acknowledged write; read-modify-write. If a read operation is specified, the Data and Mask parameters shall be invalid. If a write operation is specified, the Mask parameter shall be invalid.

Proxy_ID specifies the proxy ID for the proxy request.

Proxy_Key specifies the proxy key for the proxy request.

Service_ID specifies the ID of the service specified by the proxy request.

Parameter_ID specifies the ID of the parameter specified by the proxy request.

Entry specifies the entry index specified by the proxy request.

Field_ID specifies the ID of the field specified by the proxy request.

Length specifies the length of the operation in 32-bit words. When a read-modify-write operation is specified, this parameter shall always be 1.

Data specifies the data associated with the proxy operation in 32-bit words. This parameter shall only be valid for write and read-modify-write operations. When valid, the length of data contained in this parameter shall agree with the Length parameter. When a read-modify-write operation is specified, this parameter shall contain one 32-bit word.

Mask specifies the read-modify-write mask value as a 32-bit word. This parameter shall only be valid for read-modify-write operations.

6.2.5.23 NMS_PROXY_REPLY.request

The NMS_PROXY_REPLY.request primitive requests that a SpaceWire-PnP reply is sent, usually in response to a NMS_PROXY_REQUEST.indication primitive.

The parameters for this primitive shall be:

- RMAP_Parameters;
- Operation;
- Status;
- Length;
- Data.

RMAP_Parameters includes the source address, destination address and transaction ID.

Operation specifies the type of operation which this reply is for and shall be one of the following: read; verified acknowledged write; read-modify-write. If a write operation is specified, the Data parameter shall be invalid.

Status specifies the status value to include in the reply.

Length specifies the length of the operation in 32-bit words. When a read-modify-write operation is specified, this parameter shall always be 1.

Data specifies the data associated with the proxy operation in 32-bit words. This parameter shall only be valid for read and read-modify-write operations. When valid, the length of data contained in this parameter shall agree with the Length parameter. When a read-modify-write operation is specified, this parameter shall contain one 32-bit word.

6.2.5.24 NMS_PROXY_REPLY.indication

The NMS_PROXY_REPLY.indication primitive shall provide an indication to an initiator that a proxy reply may have been correctly sent in response to a NMS_PROXY_REPLY.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result.

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

6.2.6 Initiator Primitive Provision

Level 2 initiators shall provide level 1 primitives in accordance with the rules specified for level 1 shown in Table 5-11. In addition, level 2 initiators shall also provide primitive in accordance with Table 6-8. Where a primitive is specified both for level 1 and level 2 initiators, the provision rules for level 2 shall take precedence. “M” indicates that provision of the primitive is mandatory; “O” indicates that provision of the primitive is optional. Where an optional request primitive is provided, an indication primitive shall also be provided, and vice-versa.

Table 6-8: Level 2 Network Management Service Initiator Primitive Provision		
Primitive	**Provision**	**Notes**
NMS_DISCOVER_NETWORK.request	O	-
NMS_DISCOVER_NETWORK.indication	O	-
NMS_READ_DEVICE_OWNER.request	M	-
NMS_READ_DEVICE_OWNER.indication	M	-
NMS_MODIFY_DEVICE_OWNER.request	M	-
NMS_MODIFY_DEVICE_OWNER.indication	M	-
NMS_READ_REGION_PRIORITY.request	M	-
NMS_READ_REGION_PRIORITY.indication	M	-
NMS_READ_LOCAL_TOPOLOGY.request	M	-
NMS_READ_LOCAL_TOPOLOGY.indication	M	-
NMS_OWNER_IDENTIFICATION.request	M	-
NMS_OWNER_IDENTIFICATION.indication	M	-
NMS_CLAIM_DEVICE.request	O	-
NMS_CLAIM_DEVICE.indication	O	-
NMS_RELEASE_DEVICE.request	O	-
NMS_RELEASE_DEVICE.indication	O	-
NMS_FORCE_RELEASE.indication	O	-
NMS_GENERATE_PROXY.request	O	-
NMS_GENERATE_PROXY.indication	O	-
NMS_RELEASE_PROXY.request	O	-
NMS_RELEASE_PROXY.indication	O	-
NMS_PROXY_REQUEST.indication	O	-
NMS_PROXY_REPLY.request	O	-
NMS_PROXY_REPLY.indication	O	-

6.2.7 Timeout Determination

The device claim algorithms require the use of three timeouts values: T1, T2 and T3. These three timeouts depend on the time constants given in Table 6-9.

Table 6-9: Timeout Constants		
Constant	**Description**	**Maximum Value**
t_d	The maximum network propagation delay between transmitting the EOP of a read command packet specifying a read of 4 bytes (1 word) and receiving the EOP of the associated read reply packet.	260µs
t_p	The maximum permissible processing time between receiving the EOP of a read reply packet and transmitting the EOP of the subsequent write command packet.	100µs

A system with knowledge of the network topology and device capabilities may be able to bound t_d below the maximum value given in Table 6-9. If the system is not able to deterministically bound t_d using topological knowledge then the maximum value shall be used.

Given the timeout constants in Table 6-9, the three timeout values shall be within the bounds specified in Table 6-10.

Table 6-10: Timeout Values		
Timeout	**Minimum Value**	**Maximum Value**
T1	$8\ t_d + 2\ t_p$	∞
T2	0	t_d
T3	$4\ t_d + 2\ t_p$	$6\ t_d + 2\ t_p$

6.3 LINK CONFIGURATION SERVICE

6.3.1 Service Provision

Provision of the Link Configuration service shall be mandatory for both targets and initiators.

6.3.2 Target Parameters

Link Configuration parameters are summarised in Table 6-11.

Table 6-11: Level 2 Link Configuration Service Parameters			
ID	**Name**	**Type**	**Differences to Level 1**
0	Port Activity	Simple	None
1	Reference Rate	Simple	None
2	Link Control	Complex	None
3-7	*Reserved*	-	*Reserved for future use*

6.3.3 Target Parameter Provision

Level 2 device targets shall provide parameters in accordance with the rules defined for level 1 parameter provision (see Section 5.4.3).

6.3.4 Target Actions

A target shall not carry out any further actions beyond those specified for level 1 support (see Section 5.4.4).

6.3.5 Initiator Primitives

Level 2 initiators shall provide primitives as specified for level initiators (see Section 5.4.5).

6.3.6 Initiator Primitive Provision

Level 2 initiators shall provide primitives in accordance with the rules defined for level 1 primitive provision (see Section 5.4.6).

6.4 Router Configuration Service

6.4.1 Service Provision

Provision of the Router Configuration service shall be mandatory for all initiators and router targets only. Node targets shall not provide this service.

6.4.2 Target Parameters

Router Configuration parameters are summarised in Table 6-12.

Table 6-12: Level 2 Router Configuration Service Parameters			
ID	**Name**	**Type**	**Differences to Level 1**
0	Router Control	Simple	None
1	Routing Table	Complex	None
2-7	*Reserved*	-	*Reserved for future use*

6.4.3 Target Parameter Provision

Level 2 device targets shall provide parameters in accordance with the rules defined for level 1 parameter provision (see Section 5.5.3).

6.4.4 Target Actions

A target shall not carry out any further actions beyond those specified for level 1 support (see Section 5.5.4).

6.4.5 Initiator Primitives

Level 2 initiators shall provide primitives as specified for level initiators (see Section 5.5.5).

6.4.6 Initiator Primitive Provision

Level 2 initiators shall provide primitives in accordance with the rules defined for level 1 primitive provision (see Section 5.5.6).

6.5 Time-Code Source Service

6.5.1 Service Provision

Provision of the Time-Code Source service shall be optional for both targets and initiators.

6.5.2 Target Parameters

Level 2 Time-Code Source service parameters are summarised in Table 6-13.

Table 6-13: Level 2 Time-Code Source Service Parameters			
ID	**Name**	**Type**	**Differences to Level 1**
0	Time-Code Sources	Complex	None
1-7	*Reserved*	-	*Reserved for future use*

6.5.3 Target Parameter Provision

Level 2 device targets shall provide parameters in accordance with the rules defined for level 1 parameter provision (see Section 5.6.3).

6.5.4 Target Actions

A target shall not carry out any further actions beyond those specified for level 1 support (see Section 5.6.4).

6.5.5 Initiator Primitives

Level 2 initiators shall provide primitives as specified for level initiators (see Section 5.6.5).

6.5.6 Initiator Primitive Provision

Level 2 initiators shall provide primitives in accordance with the rules defined for level 1 primitive provision (see Section 5.6.6).

6.6 Proxy Operation and Use

This section specifies issues concerning the provision, operation and use of proxy targets not covered elsewhere in this document.

6.6.1 Proxy Target Provision

A level 2 initiator shall always provide a valid proxy target for each device it owns. The proxy target shall be valid from the instant at which the device is owned; this may require the generation of a proxy prior to the attempt to claim ownership of a device.

A proxy target shall provide access to all services, parameters and fields accessible on the target device. An owner may choose to modify parameters provided by the proxy relative to those provided by the original device, except those listed below, which the proxy shall provide without modification:

- All Device Identification service parameters

- The Network ID and Device Owner fields of the Network Identification parameter provided by the Network Management service.

Additionally, proxy targets shall provide the Region/Priority field of the Network Identification parameter and the Local Topology parameter fields (both provided by the Network Management service) as specified by Table 6-7.

A proxy shall cease to be valid only if ownership of the device is relinquished. This occurs in one of the following cases:

- The device ceases to be available (it is removed from the network or becomes inoperable).
- The device is claimed by another node. This shall be signalled to the owning initiator via the NMS_FORCE_RELEASE.indication primitive.
- Ownership of the device is relinquished by modifying the Device Owner field of the Network Identification parameter provided by the Network Management service to zero.

6.6.2 Initiator Redirection

Unless the initiator is the owner of the device, all initiator primitives shall result in an operation directed at the proxy target rather than device itself. This shall apply to all services, except for the Device Identification service which may access the device directly. If the target device has no owner, services other than the Device Identification service shall not access the device at all, unless ownership of the device is obtained.

To achieve redirection, level 2 service shall first identify the owner of the device using the algorithm described for the NMS_OWNER_IDENTIFICATION.request primitive (see Section 6.2.5.11). The target address shall be specified in relation to the device's local router, which is either:

- the device itself (if it is a router);
- the router attached to the return port identified as a result of the owner identification algorithm (if it is a node).

The target address shall then be composed of:

- the address from the initiator to the local router;
- the address from the local router to the owner (determined as a result of the owner identification algorithm).

The reply address shall be solely composed of the address from the local router to the initiator. It shall be the responsibility of the proxy target to supplement this with the return address from the proxy target to the local router.

7 CAPABILITY SERVICES

This section defines the capability services which may be supported by SpaceWire-PnP devices, defining their parameters, target actions and primitives. The capabilities a device supports are defined by the Capability List parameter of the Device Identification service (see Section 5.2.2.5). Where a capability is listed by the Capability List parameter, and a capability service is defined for that capability by the SpaceWire-PnP Protocol Definition, the device shall support the associated capability service.

7.1 CAPABILITY SERVICES OVERVIEW

SpaceWire-PnP defines support for standardised protocols, which is currently RMAP only (see [AD2]). Non-standard capability services can be provided using the prototyping protocol IDs as defined by the SpaceWire Protocols standard [AD2].

7.1.1 RMAP Capability Services

SpaceWire-PnP defines support for two RMAP capability services: Data Sources and Data Sinks. These are described by capability protocol IDs where RMAP is specified as the transport protocol. Vendors may also define RMAP capability services using IDs 240 to 253. The Protocol ID usage when RMAP is specified as a transport protocol is shown in Table 7-1.

Table 7-1: RMAP Capability Services

Transport Protocol ID	Protocol ID	Capability Service
1	0	*Reserved*
1	1	RMAP Data Source Service
1	2	RMAP Data Source Service
1	3-239	*Not yet allocated: allocated by SpaceWire Working Group*
1	240-253	*Vendor specific: parameters may be defined by vendor*
1	254-255	*Reserved*

The following sections detail each of the available capability services in turn.

7.2 COMMON CAPABILITY SERVICE PARAMETERS

All capability services shall implement a common parameter which permits the version of the capability service to be determined.

7.2.1 Target Parameters

Data Source parameters are summarised in Table 7-6.

Table 7-2: Data Source Service Parameters			
ID	**Name**	**Type**	**Summary**
0	Service Information	Simple	Provides capability service information
1-7	*Capability service specific*	-	*Defined by each capability service*

7.2.1.1 Service Information

Service Information is a simple parameter providing version information for the capability service in a standard manner.

Service Information parameter fields shall be as shown in.

Table 7-3: Target Sources Parameter Root Entry Fields		
ID	**Name**	**Summary**
0	Service Version	Count of available target data sources
1-31	*Reserved*	*Reserved for future use*

7.2.1.1.1 Service Version

The Service Version field (see Figure 7-1) shall have three parts:

- Bits 24 to 31 (where 24 is the least significant) shall hold the major version.
- Bits 16 to 23 (where 16 is the least significant) shall hold the minor version.
- Bits 8 to 15 (where 8 is the least significant) shall hold the patch number.

Bits 0 to 7 shall be reserve for future use. These bits shall be ignored when read. The version shall be meaningful in the context of the capability the service is associated with.

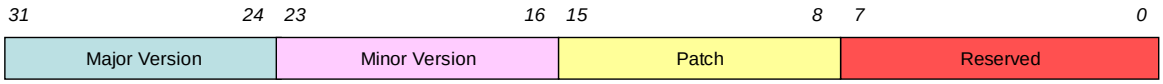


Figure 7-1: Version/Instance ID Field Encoding

7.2.2 Target Parameter Provision

Device targets shall provide parameters in accordance with the rules shown in Table 7-4. "M" indicates that provision of the parameter/field for the identified operation is mandatory; "O" indicates that provision of the parameter/field for the identified operation is optional; "F" indicates that provision of the parameter/field for the identified operation is forbidden.

Table 7-4: Common Capability Service Target Parameter Provision

Parameter	Field(s)	Read	Write	RMW
Service Information	*All*	M*	F	F

* All field values shall be constant.

7.2.3 Target Actions

A target shall not carry out any further actions beyond support for read/write/read-modify-write commands specified in Section 7.2.2.

7.2.4 Initiator Primitives

A summary of common capability service initiator primitives is presented below:

- CCS_READ_VERSION.request
- CCS_READ_VERSION.indication

These primitives are considered in more detail in the following sections.

7.2.4.1 CCS_READ_VERSION.request

The CCS_READ_VERSION.request primitive shall permit an initiator to request the version for the specified capability service.

The parameters for this primitive shall be:

- RMAP_Parameters
- Capability_Index.

RMAP_Parameters includes the source address, destination address and transaction ID.

Capability_Index the numerical index of the capability in the list provided by the DIDS_READ_CAPABILITY_LIST.indication primitive (see Section 5.2.5.10).

7.2.4.2 CCS_READ_VERSION.indication

The CCS_READ_VERSION.indication primitive shall provide an indication to an initiator that capability service version information may be available in response to a CCS_READ_VERSION.request. This primitive shall indicate whether the request was successful or not, and may contain version information.

The parameters for this primitive shall be:

- Result
- Version

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Version parameter may be invalid.

Version shall contain the version of the capability service (see Section 7.2.4.1).

7.2.5 Initiator Primitive Provision

Initiators shall provide primitives in accordance with the rules shown in Table 7-5. "M" indicates that provision of the primitive is mandatory; "O" indicates that provision of the primitive is optional. Where an optional request primitive is provided, an indication primitive shall also be provided, and vice-versa.

Table 7-5: Common Capability Service Initiator Primitive Provision

Primitive	Provision	Notes
CCS_READ_VERSION.request	M	-
CCS_READ_VERSION.indication	M	-

7.3 DATA SOURCE SERVICE

7.3.1 Target Parameters

Data Source parameters are summarised in Table 7-6.

Table 7-6: Data Source Service Parameters			
ID	**Name**	**Type**	**Summary**
0	Service Information	Simple	Provides capability service information
1	Target Sources	Complex	Provides access to zero or more data source targets
2	Initiator Sources	Complex	Provides access to zero or more data source initiators
3-7	*Reserved*	-	*Reserved for future use*

7.3.1.1 Service Information

This parameter shall be as defined by the Common Capability Service Parameters section (see Section 7.2. The major, minor and patch version fields shall be specified as 1, 0 and 0 respectively.

7.3.1.2 Target Sources

Target Sources is a complex parameter providing access to zero or more data source targets.

Fields in the root entry shall be as shown in Table 7-7.

Table 7-7: Target Sources Parameter Root Entry Fields		
ID	**Name**	**Summary**
0	Target Source Count	Count of available target data sources
1-31	*Reserved*	*Reserved for future use*

Each target is represented by a non-root entry and sources data of a single, identified type for one initiator concurrently. Concurrent access is controlled via a timed leasing mechanism. Fields in non-root entries shall be as shown in Table 7-8. There shall be as many non-root entries as there are target sources identified by the Target Source Count field. The first valid entry shall be entry 1 and valid entries shall be contiguous.

Table 7-8: Target Sources Parameter Non-Root Entry Fields		
ID	**Name**	**Summary**
0	Type/Capabilities/Status	Type of data sourced/source capabilities and status
1	Command	RMAP command details to use
2	Extended Address	Extended address to use for RMAP target reads
3	Base Memory Address	Memory address to use for RMAP target reads
4	Data Length	Maximum size of data read operation supported
5	Lease Timeout	Length of lease time on this source
6	Reply Timeout	Watchdog timeout for target replies
7-31	*Reserved*	*Reserved for future use*

7.3.1.2.1 Target Source Count Field

The Target Source Count field specifies the number of valid entries in the Target Source parameter. The count shall be represented as an unsigned number from 0 to 255 in bits 0 to 7 with the least significant bit being bit 0. The field encoding is shown in Figure 7-2. Bits 8 to 31 are reserved for future use. The value of these bits shall be ignored. This field shall be read-only and constant.

Figure 7-2: Target Source Count Field Encoding

7.3.1.2.2 Type/Capabilities/Status Field

The Type/Capabilities/Status field shall define the type of data provided by this source, the capabilities of the source and the current status of the source. The Type/Capabilities/Status field shall be encoded as shown in Figure 7-3.

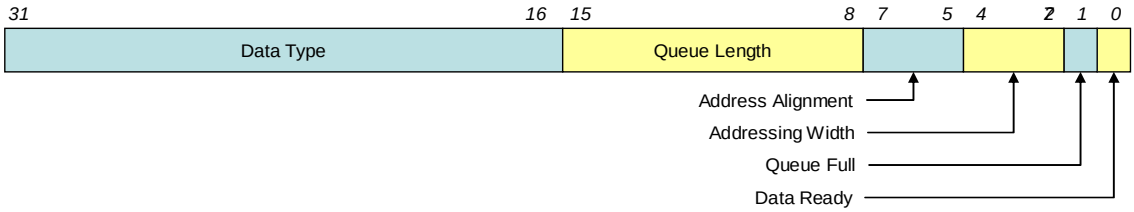


Figure 7-3 Type/Capabilities/Status Field Encoding

The field shall comprise the following parts:

- Bits 16 to 31 (where 16 is the least significant) shall identify the type of the data sourced by this data source. Valid types shall be those specified in Table 7-9.
- Bits 8 to 15 (where 8 is the least significant) shall identify the length of the read queue provided by this data source. This is the number of pending reads which may be queued at any time. Valid queue lengths shall be 0 to 255. If a queue length of 0 is specified, the reply timeout feature shall not be available.
- Bits 5 to 7 (where 5 is the least significant) shall contain the addressing width of the target. The width of each address location on the target, in bytes, is two raised to the power of the number specified.
- Bits 2 to 4 (where 2 is the least significant) shall contain the required address alignment of reads on the target. The required alignment, in bytes, is two raised to the power of the number specified. This number shall be equal or greater than the addressing width.
- Bit 1 shall indicate the status of the queue. If there is no space for further read operations remaining in the queue (or if the queue has length 0) then this bit shall be set to 1. If there is remaining space in the queue, this bit shall be set to 0.
- Bit 0 shall indicate the data availability of this data source. If there is data available to be read, this bit shall be set to 1; if there is no data available, this bit shall be set to 0.

The Type/Capabilities/Status field shall be read-only.

Valid types shall be those specified in Table 7-9. A data type of zero is reserved for future use and shall not be used. Data types from 1 to 255 are allocated and assigned by the SpaceWire Working Group and shall be used in accordance with the released list of data type identifiers and with the relevant data format specifications. The use of data types from 256-65535 shall be vendor specific.

Table 7-9: Data Type Identifiers		
Type ID	**Type**	**Defined In**
0	*Reserved*	*Reserved for future use*
1	Device Status	All three fields of the Device Status parameter
2-255	*Not Yet Allocated*	*Allocated by SpaceWire Working Group*
256-65535	*Vendor Specific*	*Data format may be chosen by the vendor*

7.3.1.2.3 Command

The Command Field shall contain the configuration of the RMAP command to be used when accessing this source, as well as the key value to use. The Command field shall be encoded as shown in Figure 7-4.

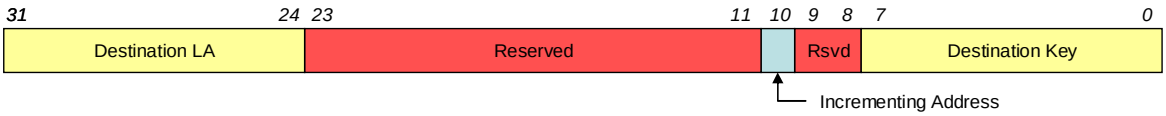


Figure 7-4: Command Field Encoding

The Command field shall have the following parts:

- Bits 24 to 31 (where 24 is the least significant) shall contain the destination logical address to be used by RMAP read commands. If this is zero then an initiator may use any destination logical address.
- Bit 10 shall indicate whether incrementing addressing mode should be used when addressing RMAP read to this source. If this bit is set to 1, incrementing addressing should be used; if this bit is set to 0, non-incrementing addressing should be used.
- Bits 0 to 7 (where 0 is the least significant) shall contain the destination key to be used when accessing this source.

Bits 11 to 23 and bits 8 and 9 shall be reserved for future use. The value of these bits shall be ignored when read and shall be written as zeros. This field shall be read-only.

7.3.1.2.4 Extended Address

The Extended Address field shall contain the extended address on the RMAP target that read commands shall use. The extended address is the upper 8 bits of the full 40-bit RMAP address. Additionally, the field shall specify whether read commands accessing this data source are permitted

to use addresses offset from the extended address/base address of this source. The Extended Address field shall be encoded as shown in Figure 7-5.

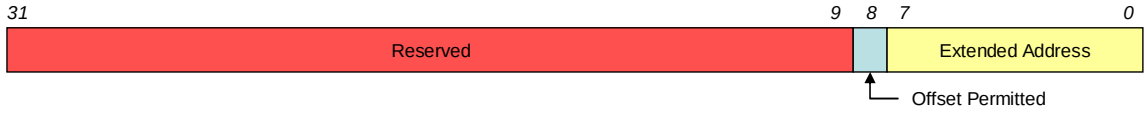


Figure 7-5: Extended Address Field Encoding

The Extended Address field shall have two parts:

- Bits 0 to 7 (where 0 is the least significant) shall contain the extended address to be used when issuing RMAP read commands on this data source
- Bit 8 shall indicate whether reads on this target are permitted with base addresses other than the address specified by Extended Address field and the Base Address field. If this bit is set to 0 the only valid base address that shall be used for reading this source is as specified by the Extended Address field and the Base Address field. If this bit is set to 1, other addresses may be used providing they fall into the valid range for this source, i.e. between the base address and the base address plus the data length. In this case the read shall not extended beyond the valid range for this source.

Bits 9 to 31 shall be reserved for future use. The value of these bits shall be ignored when read. This field shall be read-only and constant.

7.3.1.2.5 Base Memory Address

The Base Memory Address field shall contain the lowest 32 bits of the full 40-bit RMAP address that shall be used when issuing read commands on this data source. The address shall use the full 32 bits of the field, where bit 0 is the least significant. This field shall be read-only and constant.

7.3.1.2.6 Data Length Field

The Data Length field shall indicate the length of data sourced by this data source i.e. the length of individual read operations that should be directed at this target. The Data Length field shall be encoded as shown in Figure 7-6.

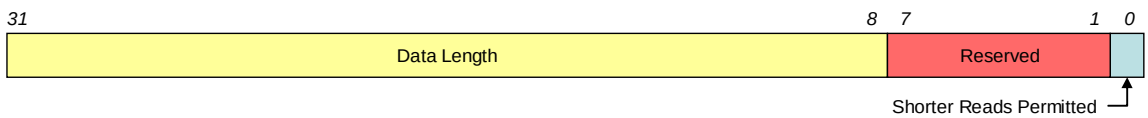


Figure 7-6: Data Length Field Encoding

The Data Length field has two parts:

- Bits 8 to 31 (where 8 is the least significant) shall specify the maximum number of bytes that can and will ever be provided by this source.
- Bit 0 shall specify whether this source permits read operations specifying lengths shorter than the maximum length. If this bit is set, read operations of lengths shorter than the maximum length are permitted; if this bit is set all read operations must be of the maximum length.

Bits 1 to 7 shall be reserved for future use. The value of these bits shall be ignored when read. The Data Length field shall be read-only.

7.3.1.2.7 Lease Timeout

The Lease Timeout field shall indicate the duration of the current lease on this source. The full 32 bits of the field (where 0 is the least significant) shall be used to specify a numeric value in microseconds. The value 0xFFFFFFFF shall be treated distinctly as infinity. Devices shall support the full range of timeouts. The lease timer shall be programmed with the value written to this field whenever it is written. The value in this field shall not count down with the lease timer but shall remain constant unless the lease timer expires (reaches zero), upon which the value in this field shall be set to zero by the source. The value of zero shall indicate that the current lease has expired and the source is available. If there are queued read commands when the lease expires, the queue shall be flushed and replies shall not be issued to the pending reads. Whenever a read command is received by this target, the lease timer shall be reset to the last value that was written.

7.3.1.2.8 Reply Timeout Field

The Reply Timeout field shall specify the maximum amount of time that shall be allowed to elapse before this source must issue a reply to a queued read. The full 32 bits of the field (where 0 is the least significant) shall be used to specify a numeric value in microseconds. The value 0xFFFFFFFF shall be treated distinctly as infinity. Devices shall support a single contiguous range of timeouts (not including 0xFFFFFFFF), all devices shall support infinite timeouts; however, devices may choose not to support the full range of timeouts discounting infinity. If a value higher than the maximum is written, the field shall be set to the maximum value; if a value lower than the minimum is written, the field shall be set to the minimum value.

7.3.1.3 Initiator Sources

Initiator Sources is a complex parameter providing access to zero or more data source initiators.

Fields in the root entry shall be as shown in Table 7-10.

Table 7-10: Initiator Sources Parameter Root Entry Fields		
ID	**Name**	**Summary**
0	Initiator Source Count	Count of available initiator data sources
1-31	*Reserved*	*Reserved for future use*

Each initiator is represented by a non-root entry and sources data of a single, identified type for one target concurrently. Concurrent access is controlled via a timed leasing mechanism. Fields in non-root entries shall be as shown in Table 7-11. There shall be as many non-root entries as there are target sources identified by the Initiator Source Count field. The first valid entry shall be entry 1 and valid entries shall be contiguous.

Table 7-11: Initiator Sources Parameter Non-Root Entry Fields		
ID	**Name**	**Summary**
0	Type/Capabilities/Status	Type of data sourced/source capabilities and status
1	Destination Address 1	First 4 bytes of destination address
2	Destination Address 2	Middle 4 bytes of destination address
3	Destination Address 3	Last 4 bytes of destination address
4	Command	RMAP command details to use
5	Source Address 1	First 4 bytes of source address
6	Source Address 2	Middle 4 bytes of source address
7	Source Address 3	Last 4 bytes of source address
8	Transaction	RMAP transaction details to use
9	Base Memory Address	Memory address to specify
10	Data Length	Maximum size of data written by this initiator
11	Lease Timeout	Length of lease time on this source
12	Reply Timeout	Length of reply timeout on this source
13-31	*Reserved*	*Reserved for future use*

7.3.1.3.1 Initiator Source Count Field

The Initiator Source Count field specifies the number of valid entries in the Initiator Source parameter. The count shall be represented as an unsigned number from 0 to 255 in bits 0 to 7 with the least significant bit being bit 0. The field encoding is shown in Figure 7-7. Bits 8 to 31 are reserved for future use. The value of these bits shall be ignored. This field shall be read-only and constant.

Figure 7-7: Initiator Source Count Field Encoding

7.3.1.3.2 Type/Capabilities/Status Field

The Type/Capabilities/Status field shall define the type of data provided by this source, the number of outstanding transactions this source may have and the current status of the source. The Type/Capabilities/Status field shall be encoded as shown in Figure 7-8.

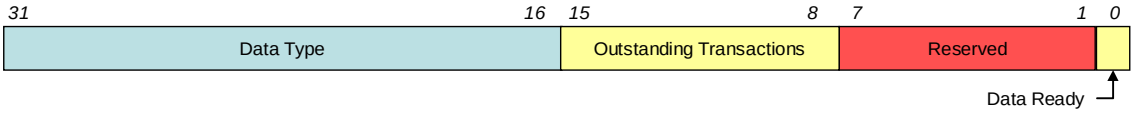


Figure 7-8 Type/Capabilities/Status Field Encoding

The field shall comprise the following parts:

- Bits 16 to 31 (where 16 is the least significant) shall identify the type of the data sourced by this data source. Valid types shall be those specified in Table 7-9.
- Bits 7 to 15 (where 7 is the least significant) shall identify the number of outstanding transactions (those awaiting acknowledgement) that this source may have.
- Bit 0 shall indicate the data availability of this data source. If there is data available to be read, this bit shall be set to 1; if there is no data available, this bit shall be set to 0.

Bits 1 to 8 shall be reserved for future use. The value of these bits shall be ignored when read. The Type/Capabilities/Status field shall be read-only.

Valid types shall be those specified in Table 7-9 (see Section 7.3.1.2.2). A data type of zero is reserved for future use and shall not be used. Data types from 1 to 255 are allocated and assigned by the SpaceWire Working Group and shall be used in accordance with the released list of data type identifiers and with the relevant data format specifications. The use of data types from 256-65535 shall be vendor specific.

7.3.1.3.3 Destination Address Fields

The Destination Address fields (1-3) encode the destination path address for write operations initiated by this source. If a path address is not required, and a single logical address is sufficient, all three of these fields shall be zero. The default value of these fields shall be also zero.

Each field shall be split into four 1-byte parts, as shown in Figure 7-9. Path addresses shall be packed into these bytes by placing the last byte of the path address into byte 4 of Destination Address field 3, the second to last byte into byte 3 and so on. All unused bytes in all Destination Address fields should be set to 0.

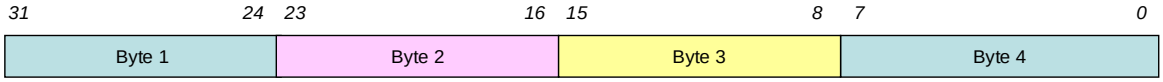


Figure 7-9: Destination Address Field Encoding

Table 7-12 shows examples of how path addresses shall be encoded into the Destination Address fields.

Table 7-12: Destination Path Address Encoding Examples

Destination Address Fields			**Resulting Path Address**
1	**2**	**3**	
0x00000000	0x00000000	0x00000000	0x00
0x00000000	0x00000000	0x00000102	0x01 0x02
0x00000000	0x00000000	0x00010002	0x01 0x00 0x02
0x00000000	0x00000000	0x00010200	0x01 0x02 0x00
0x00000000	0x00000000	0x01020304	0x01 0x02 0x03 0x04
0x00000000	0x00000001	0x02030405	0x01 0x02 0x03 0x04 0x05
0x00000001	0x02030405	0x06070809	0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09

7.3.1.3.4 Command Field

The Command Field shall contain the destination logical address, the source logical address, the key and the configuration of the RMAP command to be generated. The Command field shall be encoded as shown in Figure 7-10.

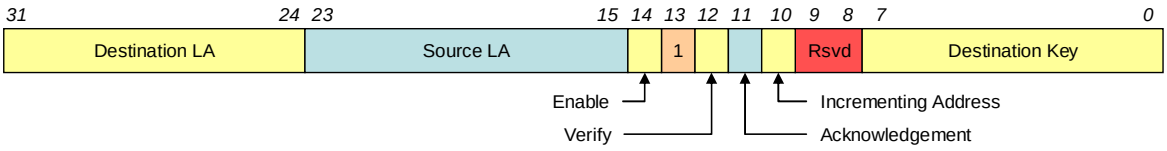


Figure 7-10: Command Field Encoding

The Command field shall have the following parts:

- Bits 24 to 31 (where 24 is the least significant) shall contain the destination logical address to be used by the RMAP write command.

- Bits 15 to 23 (where 15 is the least significant) shall contain the source logical address to be used by the RMAP write command.
- Bit 14 shall enable the initiator associated with this data source. When this bit is set to 1, the initiator shall generate RMAP write commands containing source data. When this bit is set to 0, no commands shall be initiated.
- Bit 13 shall always be 1.
- Bit 12 shall indicate if a verified write command should be generated by this initiator. Setting this bit to a 1 shall generate a verified write; setting this bit to a 0 shall generate an unverified write.
- Bit 11 shall indicate whether an RMAP write should be generated requesting an acknowledgement. If this bit is set to 1, the initiated RMAP write shall request an acknowledgement; if this bit is set to 0, the initiated RMAP write shall not request and acknowledgement. Acknowledgements (replies) may be discarded by the device without any processing.
- Bit 10 shall indicate whether an RMAP write should be generated requesting incrementing addressing. If this bit is set to 1, the initiated RMAP write shall request incrementing addressing; if this bit is set to 0, the initiated RMAP write shall request non-incrementing addressing.
- Bits 0 to 7 (where 0 is the least significant) shall contain the destination key that shall be used by RMAP writes initiated by this source.

Bits 8 and 9 shall be reserved for future use. The value of these bits shall be ignored when read and shall be written as zeros.

7.3.1.3.5 Source Address Fields

The Source Address fields shall encode the source path address (the return address) in an identical manner to the encoding used for the Destination Address fields (see Section 7.3.1.3.3). If a source path address is not required, these fields shall be set to zero. The default value for the Source Address fields shall be zero.

7.3.1.3.6 Transaction Field

The Transaction field shall contain the source logical address, the highest 8 bits of the transaction identifier and the highest 8 bits of the 40-bit base address for the RMAP write commands initiated by this source. The Transaction field shall be encoded as shown in Figure 7-11.

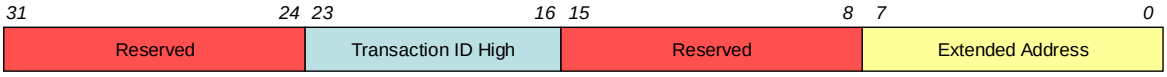


Figure 7-11: Transaction Field Encoding

The Transaction field shall have two parts:

- Bits 15 to 23 (where 15 is the least significant) shall contain the highest 8 bits of the transaction identifier to be used by the RMAP write command. The entry number for this source shall be used as the lowest 8 bits of the transaction identifier.
- Bits 0 to 7 (where 0 is the least significant) shall contain the extended address to be used by the RMAP write command. This is the top 8 bits of the full 40-bit base memory address.

Bits 8 to 15 and 24 to 31 shall be reserved; these bits shall be ignored when read and shall be written as zero.

7.3.1.3.7 Base Memory Address Field

The Base Memory Address field shall contain the lowest 32 bits of the full 40-bit base memory address to be used by the RMAP write command. Bit 0 shall be the least significant bit.

7.3.1.3.8 Data Length Field

The Data Length field shall indicate the length of data sourced by this data sourced i.e. the length of individual write operations that will be initiated by this source. The Data Length field shall be encoded as shown in Figure 7-12.

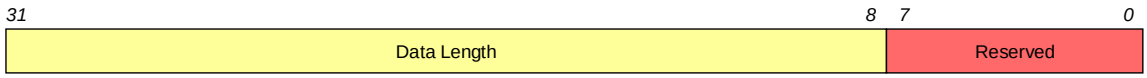


Figure 7-12: Data Length Field Encoding

Bits 8 to 31 (where 8 is the least significant) shall specify the number of bytes that will be written by this source. Bits 0 to 7 shall be reserved for future use. The value of these bits shall be ignored when read. The Data Length field shall be constant and read-only.

7.3.1.3.9 Lease Timeout Field

The Lease Timeout field shall indicate the duration of the current lease on this source. The full 32 bits of the field (where 0 is the least significant) shall be used to specify a numeric value in microseconds. The value 0xFFFFFFFF shall be treated distinctly as infinity. Devices shall support the full range of timeouts. The lease timer shall be programmed with the value written to this field whenever it is

written. The value in this field shall not count down with the lease timer but shall remain constant unless the lease timer expires (reaches zero), upon which the value in this field shall be set to zero by the source. The value of zero shall indicate that the current lease has expired and the source is available. If there are queued read commands when the lease expires, the queue shall be flushed and replies shall not be issued to the pending reads. Whenever a read command is received by this target, the lease timer shall be reset to the last value that was written.

7.3.1.3.10 Reply Timeout Field

The Reply Timeout field shall specify the maximum amount of time that shall be allowed to elapse before this source must issue a reply to an initiated acknowledged write which has not yet received an acknowledgement. The full 32 bits of the field (where 0 is the least significant) shall be used to specify a numeric value in microseconds. The value 0xFFFFFFFF shall be treated distinctly as infinity. Devices shall support a single contiguous range of timeouts (not including 0xFFFFFFFF), all devices shall support infinite timeouts; however, devices may choose not to support the full range of timeouts discounting infinity. If a value higher than the maximum is written, the field shall be set to the maximum value; if a value lower than the minimum is written, the field shall be set to the minimum value.

7.3.2 Target Parameter Provision

Device targets shall provide parameters in accordance with the rules shown in Table 7-13. “M” indicates that provision of the parameter/field for the identified operation is mandatory; “O” indicates that provision of the parameter/field for the identified operation is optional; “F” indicates that provision of the parameter/field for the identified operation is forbidden.

Table 7-13: Data Source Capability Service Target Parameter Provision				
Parameter	**Field(s)**	**Read**	**Write**	**RMW**
Service Information	*All fields*	M*	F	F
Target Sources	*Root entry fields*	M*	F	F
Target Sources	Type/Capabilities/Status	M	F	F
Target Sources	Command, Data Length, Extended Address/Mode, Base Memory Addresss	M*	F	F
Target Sources	Lease Timeout, Reply Timeout	M	M	O
Initiator Sources	*Root entry fields*	M*	F	F
Initiator Sources	Type/Capabilities/Status	M	F	F
Initiator Sources	*All other fields*	M	M	O

* All field values shall be constant.

7.3.3 Target Actions

The Data Source capability service provides configuration for a data source which provides data via RMAP. The target actions specified in this section define the interaction between the SpaceWire-PnP capability service and the RMAP data source, acting either in the capacity of an RMAP target or initiator or both.

Data Source Service targets shall implement the Target Data Available action. Data Source initiators shall implement the Initiator Data Available action.

7.3.3.1 Target Data Available

The Target Data Available action comprises the following algorithms:

When an RMAP read command is received by the RMAP target, the following algorithm shall be used:

1. If the queue length is zero, a reply shall be issued immediately.
    a. If data is not available the reply shall not contain data and shall specify the error “RMAP Command not implemented or not authorised”.
    b. If data is available the reply shall contain the data, as requested by the command.
2. If the queue length is non-zero the reply will be delayed (unless the queue is full).

a. If the queue is full a reply shall be issued immediately specifying the error “RMAP Command not implemented or not authorised”.

b. If the queue is empty, and data is available, a reply shall be issued immediately containing the data as requested by the command.

c. If the queue is not full and data is not available, the read request shall be queued pending data and the reply timeout (if configured) shall be started.

When data becomes available the following algorithm shall be used:

1. If there is a pending read in the queue then a reply shall be issued.
2. If there is no pending read in the queue, and the lease timeout is non-zero, then the data shall be held as available (pending transmission).
3. If the lease timeout is zero then the data shall be discarded.

When the reply timeout expires (becomes zero) the following algorithm shall be used:

1. If there is a pending read in the queue then a reply shall be issued containing the most recent version of the data, if possible, or specifying the error “RMAP Command not implemented or not authorised”, if no recent data is available. The reply timeout shall then be restarted.
2. If there is no pending read then the reply timeout shall be allowed to remain expired (at zero).

When the lease timeout expires (becomes zero) the following algorithm shall be used:

1. All available pending data shall be discarded.
2. Any queued reads shall be flushed and no replies shall be generated.

7.3.3.2 Initiator Data Available

The Initiator Data Available action comprises the following algorithms:

When data becomes available the following algorithm shall be used:

1. If the lease timeout is non-zero and the initiator has the capacity for additional outstanding transactions then a write command shall be initiated as specified by the various fields of the relevant Initiator Sources parameter entry. The data shall be held as pending acknowledgement. The reply timeout shall be started.
2. If the lease timeout is zero or the initiator does not have the capacity for additional outstanding transactions then the data shall be discarded.

When a reply is received which matches the transaction ID of the last command to be sent, or the reply timeout field is written to, the following algorithm shall be used:

1. If there is data held as pending acknowledgement then the data shall discarded as an acknowledgement has been received.

When the reply timeout expires (reaches zero) then the following algorithm shall be used:

1. If there is data held as pending acknowledgement then the write command shall be re-initiated as specified by the various fields of the relevant Initiator Sources parameter entry. The data shall continue to be held as pending acknowledgement. The reply timeout shall be restarted.
2. If there is no data held as pending then the reply timeout shall be allowed to remain expired (at zero).

When the lease timeout expires (reaches zero) then the following algorithm shall be used:

1. If there is data held as pending acknowledgement then the data shall be discarded.

7.3.4 Initiator Primitives

A summary of Data Source service initiator primitives is presented below:

- SRC_GET_TARGET_COUNT.request
- SRC_GET_TARGET_COUNT.indication
- SRC_READ_TARGET_CONFIG.request
- SRC_READ_TARGET_CONFIG.indication
- SRC_WRITE_TARGET_CONFIG.request
- SRC_WRITE_TARGET_CONFIG.indication
- SRC_READ_TARGET_DATA.request
- SRC_READ_TARGET_DATA.indication
- SRC_GET_INITIATOR_COUNT.request
- SRC_GET_INITIATOR_COUNT.indication
- SRC_READ_INITIATOR_CONFIG.request
- SRC_READ_INITIATOR_CONFIG.indication
- SRC_WRITE_INITIATOR_CONFIG.request
- SRC_WRITE_INITIATOR_CONFIG.indication
- SRC_INITIATOR_DATA_READY.indication

- SRC_ACK_INITIATOR.request
- SRC_ACK_INITIATOR.indication

These primitives are considered in more detail in the following sections.

7.3.4.1 SRC_GET_TARGET_COUNT.request

The SRC_GET_TARGET_COUNT.request primitive shall permit an initiator to request the count of target sources provided by the capability service.

The parameters for this primitive shall be:

- RMAP_Parameters
- Capability_Index

RMAP_Parameters includes the source address, destination address and transaction ID.

Capability_Index the numerical index of the data source capability service in the list provided by the DIDS_READ_CAPABILITY_LIST.indication primitive (see Section 5.2.5.10).

7.3.4.2 SRC_GET_TARGET_COUNT.indication

The SRC_GET_TARGET_COUNT.indication primitive shall provide an indication to an initiator that a target data source count may be available in response to a SRC_GET_TARGET_COUNT.request. This primitive shall indicate whether the request was successful or not, and may contain a data source target count.

The parameters for this primitive shall be:

- Result
- Target_Count

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Target_Count parameter may be invalid.

Target_Count shall contain the count of target sources provided by this capability service (see Section 7.3.1.2.1).

7.3.4.3 SRC_READ_TARGET_CONFIG.request

The SRC_READ_TARGET_CONFIG.request primitive shall permit an initiator to request the current configuration of a data source target.

The parameters for this primitive shall be:

- RMAP_Parameters
- Capability_Index
- Target_Index

RMAP_Parameters includes the source address, destination address and transaction ID.

Capability_Index the numerical index of the data source capability service in the list provided by the DIDS_READ_CAPABILITY_LIST.indication primitive (see Section 5.2.5.10).

Target_Index shall indicate the numerical index of the data source target provided by the capability service.

7.3.4.4 SRC_READ_TARGET_CONFIG.indication

The SRC_READ_TARGET_CONFIG.indication primitive shall provide an indication to an initiator that configuration information for a target data source may be available in response to a SRC_READ_TARGET_CONFIG.request. This primitive shall indicate whether the request was successful or not, and may contain data source target configuration.

The parameters for this primitive shall be:

- Result
- Data_Type
- Queue_Length
- Queue_Full
- Data_Ready
- Address_Alignment
- Address_Width
- Destination_LA
- Destination_Key
- Use_Incrementing
- Extended_Address
- Base_Address
- Offset_OK
- Data_Length

- Shorter_OK
- Lease_Timeout
- Reply_Timeout

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the all other parameters may be invalid.

Data_Type shall indicate the type of data provided by this data source (see Section 7.3.1.2.2).

Queue_Length shall indicate the maximum length of the data source queue (see Section 7.3.1.2.2).

Queue_Full shall indicate whether or not the queue is currently full (see Section 7.3.1.2.2).

Data_Ready shall indicate whether or not there is currently data available (see Section 7.3.1.2.2).

Address_Alignment shall indicate the addressing alignment required by the source (see Section 7.3.1.2.2).

Address_Width shall indicate the addressing width used by the source (see Section 7.3.1.2.2).

Destination_LA shall indicate the destination logical address value to use when accessing the source, or zero if any logical address can be used (see Section 7.3.1.2.3).

Destination_Key shall indicate the destination key value to use when accessing the source (see Section 7.3.1.2.3).

Use_Incrementing shall indicate whether or not reads addressed to the data source should use incrementing addressing mode (see Section 7.3.1.2.3).

Extended_Address shall indicate the extended address to use when accessing the source (see Section 7.3.1.2.3).

Base_Address shall indicate the base address to use when accessing the source (see Section 7.3.1.2.3).

Offset_OK shall indicate whether or not the target accepts reads which are offset from the extended address/base address specified by the Extended_Address and Base_Address parameters (see Section 7.3.1.2.3).

Data_Length shall indicate the length of data provided by the source (see Section 7.3.1.2.6).

Shorter_OK shall indicate whether or not the target accepts reads which are shorter than the length specified by the Data_length parameter (see Section 7.3.1.2.6).

Lease_Timeout shall indicate the length of the current lease timeout, or zero if the source is not currently leased (see Section 7.3.1.2.7).

Reply_Timeout shall indicate the length of the current reply timeout, or zero if no reply timeout is required (see Section 7.3.1.2.8).

7.3.4.5 SRC_WRITE_TARGET_CONFIG.request

The SRC_WRITE_TARGET_CONFIG.request primitive shall permit an initiator to set the current configuration of a data source target.

The parameters for this primitive shall be:

- RMAP_Parameters
- Capability_Index
- Target_Index
- Lease_Timeout
- Reply_Timeout

RMAP_Parameters includes the source address, destination address and transaction ID.

Capability_Index the numerical index of the data source capability service in the list provided by the DIDS_READ_CAPABILITY_LIST.indication primitive (see Section 5.2.5.10).

Target_Index the numerical index of the data source target provided by the capability service.

Lease_Timeout shall indicate the length of the desired lease timeout, or zero if the source is not to be leased (see Section 7.3.1.2.6).

Reply_Timeout shall indicate the length of the current reply timeout, or zero if no reply timeout is required (see Section 7.3.1.2.8).

7.3.4.6 SRC_WRITE_TARGET_CONFIG.indication

The SRC_WRITE_TARGET_CONFIG.indication primitive shall provide an indication to an initiator that configuration information for a target data source may have been written in response to a SRC_WRITE_TARGET_CONFIG.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

7.3.4.7 SRC_READ_TARGET_DATA.request

The SRC_READ_TARGET_DATA.request primitive shall permit an initiator to issue an RMAP read to retrieve data from a target data source, if it is available.

The parameters for this primitive shall be:

- RMAP_Parameters
- Destination_Key
- Incrementing_Read
- Extended_Address
- Base_Address
- Data_Length

RMAP_Parameters includes the source address, destination address and transaction ID.

Destination_Key shall indicate the destination key to use for the RMAP read operation.

Incrementing_Read shall indicate whether the RMAP read operation should use an incrementing read or not.

Extended_Address shall indicate the extended address to use for the RMAP read operation.

Base_Address shall indicate the base address to use for the RMAP read operation.

Data_Length shall indicate the data length to request in the RMAP read operation.

7.3.4.8 SRC_READ_TARGET_DATA.indication

The SRC_READ_TARGET_DATA.indication primitive shall provide an indication to an initiator that target source data may be available in response to a SRC_READ_TARGET_DATA.request. This primitive shall indicate whether the request was successful or not, and may contain data.

The parameters for this primitive shall be:

- Result
- Transaction_ID
- Data
- Data_Length

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then other parameters may be invalid.

Transaction_ID shall indicate the transaction ID of the RMAP read operation which has completed (as supplied to the SRC_READ_TARGET_DATA.request primitive).

Data shall contain the data return by the RMAP read reply, if any.

Data_Length shall indicate the length of data available in the Data parameter.

7.3.4.9 SRC_GET_INITIATOR_COUNT.request

The SRC_GET_INITIATOR_COUNT.request primitive shall permit an initiator to request the count of initiator sources provided by the capability service.

The parameters for this primitive shall be:

- RMAP_Parameters
- Capability_Index

RMAP_Parameters includes the source address, destination address and transaction ID.

Capability_Index the numerical index of the data source capability service in the list provided by the DIDS_READ_CAPABILITY_LIST.indication primitive (see Section 5.2.5.10).

7.3.4.10 SRC_GET_INITIATOR_COUNT.indication

The SRC_GET_INITIATOR_COUNT.indication primitive shall provide an indication to an initiator that an initiator data source count may be available in response to a SRC_GET_INITIATOR_COUNT.request. This primitive shall indicate whether the request was successful or not, and may contain a data source initiator count.

The parameters for this primitive shall be:

- Result
- Initiator_Count

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Target_Count parameter may be invalid.

Initiator_Count shall contain the count of initiator sources provided by this capability service (see Section 7.3.1.3.10).

Reply_Timeout shall indicate the length of the current reply timeout, or zero if no reply timeout is in use (see Section 7.3.1.2.8).

7.3.4.11 SRC_READ_INITIATOR_CONFIG.request

The SRC_READ_INITIATOR_CONFIG.request primitive shall permit an initiator to request the current configuration of a data source initiator.

The parameters for this primitive shall be:

- RMAP_Parameters
- Capability_Index
- Initiator_Index

RMAP_Parameters includes the source address, destination address and transaction ID.

Capability_Index the numerical index of the data source capability service in the list provided by the DIDS_READ_CAPABILITY_LIST.indication primitive (see Section 5.2.5.10).

Initiator_Index shall indicate the numerical index of the data source initiator provided by the capability service.

7.3.4.12 SRC_READ_INITIATOR_CONFIG.indication

The SRC_READ_INITIATOR_CONFIG.indication primitive shall provide an indication to an initiator that configuration information for an initiator data source may be available in response to a SRC_READ_INITIATOR_CONFIG.request. This primitive shall indicate whether the request was successful or not, and may contain data source initiator configuration.

The parameters for this primitive shall be:

- Result
- Data_Type
- Outstanding_Transactions
- Data_Ready
- Enabled
- Destination_Path
- Destination_LA
- Destination_Key
- Verify

- Acknowledge
- Use_Incrementing
- Source_Path
- Source_LA
- Transaction_ID
- Extended_Address
- Base_Address
- Data_Length
- Lease_Timeout
- Reply_Timeout

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the all other parameters may be invalid.

Data_Type shall indicate the type of data provided by this data source (see Section 7.3.1.3.2).

Outstanding_Transactions shall indicate the maximum number of initiated writes awaiting acknowledgements which may be handled by the data source (see Section 7.3.1.3.2).

Data_Ready shall indicate whether or not there is currently data available (see Section 7.3.1.3.2).

Enabled shall indicate whether or not the initiator source is currently enabled (see Section 7.3.1.3.4).

Destination_Path shall contain the path address portion (i.e. excluding the final logical address) of the destination address; this may be empty (see Section 7.3.1.3.3).

Destination_LA shall contain the logical address of the destination (see Section 7.3.1.3.4).

Destination_Key shall indicate the destination key value to use when initiating writes from the source (see Section 7.3.1.3.4).

Verify shall indicate whether or not the source initiator should initiate verified writes (see Section 7.3.1.3.4).

Acknowledge shall indicate whether or not the source initiator should initiate acknowledged writes (see Section 7.3.1.3.4).

Use_Incrementing shall indicate whether or not the source initiator should initiate incrementing writes (see Section 7.3.1.3.4).

Source_Path shall contain the path address portion (i.e. excluding the final logical address) of the source address; this may be empty (see Section 7.3.1.3.5).

Source_LA shall contain the logical address of the source (see Section 7.3.1.3.4).

Transaction_ID shall contain the upper byte of the transaction ID used by writes initiated from this source (see Section 7.3.1.3.6).

Extended_Address shall indicate the extended address used by writes initiated from this source (see Section 7.3.1.3.6).

Base_Address shall indicate the base address used by writes initiated from this source (see Section 7.3.1.3.7).

Data_Length shall indicate the length of data provided by writes initiated from this source (see Section 7.3.1.3.8)

Lease_Timeout shall indicate the length of the current lease timeout, or zero if the source is not currently leased (see Section 7.3.1.3.9).

Reply_Timeout shall indicate the length of the current reply timeout, or zero if the reply timeout is not currently in use (see Section 7.3.1.3.10).

7.3.4.13 SRC_WRITE_INITIATOR_CONFIG.request

The SRC_WRITE_INITIATOR_CONFIG.request primitive shall permit an initiator to set the current configuration of a data source initiator.

The parameters for this primitive shall be:

- RMAP_Parameters
- Capability_Index
- Target_Index
- Enabled
- Destination_Path
- Destination_LA
- Destination_Key
- Verify
- Acknowledge
- Use_Incrementing

- Source_Path
- Source_LA
- Transaction_ID
- Extended_Address
- Base_Address
- Lease_Timeout
- Reply_Timeout

RMAP_Parameters includes the source address, destination address and transaction ID.

Capability_Index the numerical index of the data source capability service in the list provided by the DIDS_READ_CAPABILITY_LIST.indication primitive (see Section 5.2.5.10).

Target_Index the numerical index of the data source initiator provided by the capability service.

Enabled shall indicate whether or not the initiator source should be enabled (see Section 7.3.1.3.4).

Destination_Path shall contain the path address portion (i.e. excluding the final logical address) of the destination address; this may be empty (see Section 7.3.1.3.3).

Destination_LA shall contain the logical address of the destination (see Section 7.3.1.3.4).

Destination_Key shall indicate the destination key value to use when initiating writes from the source (see Section 7.3.1.3.4).

Verify shall indicate whether or not the source initiator should initiate verified writes (see Section 7.3.1.3.4).

Acknowledge shall indicate whether or not the source initiator should initiate acknowledged writes (see Section 7.3.1.3.4).

Use_Incrementing shall indicate whether or not the source initiator should initiate incrementing writes (see Section 7.3.1.3.4).

Source_Path shall contain the path address portion (i.e. excluding the final logical address) of the source address; this may be empty (see Section 7.3.1.3.5).

Source_LA shall contain the logical address of the source (see Section 7.3.1.3.4).

Transaction_ID shall contain the upper byte of the transaction ID to be used by writes initiated from this source (see Section 7.3.1.3.6).

Extended_Address shall indicate the extended address to be used by writes initiated from this source (see Section 7.3.1.3.6).

Base_Address shall indicate the base address to be used by writes initiated from this source (see Section 7.3.1.3.7).

Lease_Timeout shall indicate the length of the desired lease timeout, or zero if the source is not to be leased (see Section 7.3.1.3.9).

Reply_Timeout shall indicate the length of the desired reply timeout, or zero if the reply timeout is not required (see Section 7.3.1.3.10).

7.3.4.14 SRC_WRITE_INITIATOR_CONFIG.indication

The SRC_WRITE_INITIATOR_CONFIG.indication primitive shall provide an indication to an initiator that configuration information for an initiator data source may have been written in response to a SRC_WRITE_INITIATOR_CONFIG.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

7.3.4.15 SRC_INITIATOR_DATA_READY.indication

The SRC_INITIATOR_DATA_READY.indication primitive shall provide an indication to an initiator that data is ready due to the receipt of an initiated write from an initiator data source.

The parameters for this primitive shall be:

- Transaction_ID
- Data
- Data_Length

Transaction_ID shall indicate the full 16-bit transaction ID of the received write.

Data shall contain the data received as part of the write.

Data_Length shall indicate the length of data available in the Data parameter.

7.3.4.16 SRC_ACK_INITIATOR.request

The SRC_ACK_INITIATOR.request primitive shall permit an initiator to request the transmission of an acknowledgement in response to the receipt of data (indicated by a SRC_INITIATOR_DATA_READY.indication primitive).

The parameters for this primitive shall be:

- RMAP_Parameters
- Status

RMAP_Parameters includes the source address, destination address and transaction ID.

Status shall indicate the RMAP status code to include in the acknowledgement.

7.3.4.17 SRC_ACK_INITIATOR.indication

The SRC_ACK_INITIATOR.indication primitive shall provide an indication to an initiator that an acknowledgement to an initiator data source may have been sent in response to a SRC_ACK_INITIATOR.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

7.3.5 Initiator Primitive Provision

Initiators shall provide primitives in accordance with the rules shown in Table 7-14. “M” indicates that provision of the primitive is mandatory; “O” indicates that provision of the primitive is optional. Where an optional request primitive is provided, an indication primitive shall also be provided, and vice-versa.

Table 7-14: Data Source Capability Service Initiator Primitive Provision		
Primitive	**Provision**	**Notes**
SRC_GET_TARGET_COUNT.request	M	-
SRC_GET_TARGET_COUNT.indication	M	-
SRC_READ_TARGET_CONFIG.request	M	-
SRC_READ_TARGET_CONFIG.indication	M	-
SRC_WRITE_TARGET_CONFIG.request	M	-
SRC_WRITE_TARGET_CONFIG.indication	M	-
SRC_READ_TARGET_DATA.request	M	-
SRC_READ_TARGET_DATA.indication	M	-
SRC_GET_INITIATOR_COUNT.request	M	-
SRC_GET_INITIATOR_COUNT.indication	M	-
SRC_READ_INITIATOR_CONFIG.request	M	-
SRC_READ_INITIATOR_CONFIG.indication	M	-
SRC_WRITE_INITIATOR_CONFIG.request	M	-
SRC_WRITE_INITIATOR_CONFIG.indication	M	-
SRC_INITIATOR_DATA_READY.indication	M	-
SRC_ACK_INITIATOR.request	M	-
SRC_ACK_INITIATOR.indication	M	-

7.4 Data Sink Service

7.4.1 Target Parameters

Data Sink parameters are summarised in Table 7-15.

Table 7-15: Data Sink Service Parameters			
ID	**Name**	**Type**	**Summary**
0	Service Information	Simple	Provides capability service information
1	Target Sinks	Complex	Provides access to zero or more data sink targets
2	Initiator Sinks	Complex	Provides access to zero or more data sink initiators
3-7	*Reserved*	-	*Reserved for future use*

7.4.1.1 Service Information

This parameter shall be as defined by the Common Capability Service Parameters section (see Section 7.2. The major, minor and patch version fields shall be specified as 1, 0 and 0 respectively.

7.4.1.2 Target Sinks

Target Sinks is a complex parameter providing access to zero or more data sink targets.

Fields in the root entry shall be as shown in Table 7-16.

Table 7-16: Target Sinks Parameter Root Entry Fields		
ID	**Name**	**Summary**
0	Target Sink Count	Count of available target data sinks
1-255	*Reserved*	*Reserved for future use*

Each target is represented by a non-root entry and sinks data of a single, identified type for one initiator concurrently. Concurrent access is controlled via a timed leasing mechanism. Fields in non-root entries shall be as shown in Table 7-17. There shall be as many non-root entries as there are target sinks identified by the Target Sink Count field. The first valid entry shall be entry 1 and valid entries shall be contiguous.

Table 7-17: Target Sinks Parameter Non-Root Entry Fields		
ID	**Name**	**Summary**
0	Type/Capabilities/Status	Type of data sunk/sink capabilities and status
1	Command	RMAP command details to use
2	Extended Address	Extended address to use for RMAP target writes
3	Base Memory Address	Memory address to use for RMAP target writes
4	Data Length	Maximum size of data write operation supported
5	Lease Timeout	Length of the lease of this sink
6	Reply Timeout	Watchdog timeout for target replies
7-31	*Reserved*	*Reserved for future use*

7.4.1.2.1 Target Sink Count Field

The Target Sink Count field specifies the number of valid entries in the Target Sink parameter. The count shall be represented as an unsigned number from 0 to 255 in bits 0 to 7 with the least significant bit being bit 0. The field encoding is shown in Figure 7-13. Bits 8 to 31 are reserved for future use. The value of these bits shall be ignored. This field shall be read-only and constant.

Figure 7-13: Target Sink Count Field Encoding

7.4.1.2.2 Type/Capabilities/Status Field

The Type/Capabilities/Status field shall define the type of data expected by this sink, the capabilities of the sink and the current status of the sink. The Type/Capabilities/Status field shall be encoded as shown in Figure 7-14.

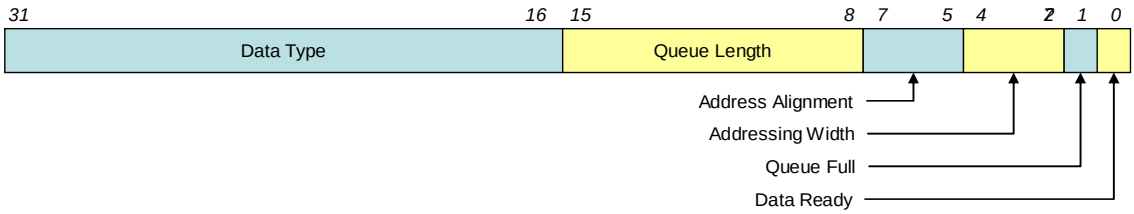


Figure 7-14 Type/Capabilities/Status Field Encoding

The field shall comprise the following parts:

- Bits 16 to 31 (where 16 is the least significant) shall identify the type of the data expected by this data sink. Valid types shall be those specified in Table 7-9.
- Bits 8 to 15 (where 8 is the least significant) shall identify the length of the write queue provided by this data sink. This is the number of pending writes which may be queued at any time. Valid queue lengths shall be 0 to 255. If a queue length of 0 is specified, the reply timeout feature shall not be available.
- Bits 5 to 7 (where 5 is the least significant) shall contain the addressing width of the target. The width of each address location on the target, in bytes, is two raised to the power of the number specified.
- Bits 2 to 4 (where 2 is the least significant) shall contain the required address alignment of writes on the target. The required alignment, in bytes, is two raised to the power of the number specified. This number shall be equal or greater than the addressing width.
- Bit 1 shall indicate the status of the queue. If there is no space for further write operations remaining in the queue (or if the queue has length 0) then this bit shall be set to 1. If there is remaining space in the queue, this bit shall be set to 0.
- Bit 0 shall indicate the data availability of this data sink. If the sink is ready to accept data immediately, this bit shall be set to 1; if the sink cannot accept data immediately this bit shall be set to 0.

The Type/Capabilities/Status field shall be read-only.

Valid types shall be those specified in Table 7-9 (see Section 7.3.1.2.2). A data type of zero is reserved for future use and shall not be used. Data types from 1 to 255 are allocated and assigned by the SpaceWire Working Group and shall be used in accordance with the released list of data type identifiers and with the relevant data format specifications. The use of data types from 256-65535 shall be vendor specific.

7.4.1.2.3 Command

The Command Field shall contain the destination logical address, the source logical address, the key and the configuration of the RMAP command to be generated. The Command field shall be encoded as shown in Figure 7-10.

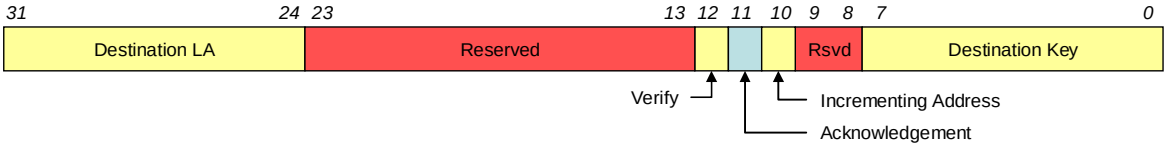


Figure 7-15: Command Field Encoding

The Command field shall have the following parts:

- Bits 24 to 31 (where 24 is the least significant) shall contain the destination logical address to be used by RMAP write commands. If this is zero then an initiator may use any destination logical address.
- Bit 12 shall indicate if a verified write command should be used by RMAP write commands addressed to this target. If this bit is 1 a verified write shall be used; if this bit is 0 an unverified write shall be used.
- Bit 11 shall indicate if an acknowledged write should be used by RMAP write commands addressed to this target. If this bit is set to 1, an acknowledged write should be used; if this bit is set to 0, an unacknowledged write should be used.
- Bit 10 shall indicate whether an incrementing write should be used by RMAP write commands addressed to this target. If this bit is set to 1, a write command specifying incrementing addressing should be used; if this bit is set to 0, the a non-incrementing write should be used.
- Bits 0 to 7 (where 0 is the least significant) shall contain the destination key that shall be used by RMAP writes initiated by this source.

Bits 8 and 9 and bits 13 to 23 shall be reserved for future use. The value of these bits shall be ignored when read and shall be written as zeros. This field shall be constant and read-only.

7.4.1.2.4 Extended Address

The Extended Address field shall contain the extended address on the RMAP target that write commands shall use. The extended address is the upper 8 bits of the full 40-bit RMAP address. The Extended Address field shall be encoded as shown in Figure 7-16.

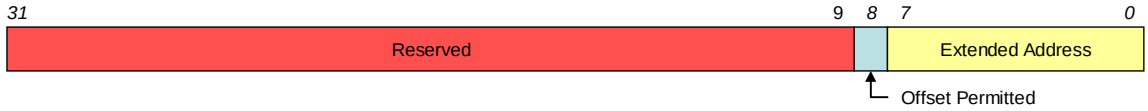


Figure 7-16: Extended Address/Mode Field Encoding

The Extended Address field shall have two parts:

- Bits 0 to 7 (where 0 is the least significant) shall contain the extended address to be used when issuing RMAP write commands on this data sink
- Bit 8 shall indicate whether writes to this target are permitted with base addresses other than the address specified by Extended Address field and the Base Address field. If this bit is set to 0 the only valid base address that shall be used for writing to this source is as specified by the Extended Address field and the Base Address field. If this bit is set to 1, other addresses may be used providing they fall into the valid range for this source, i.e. between the base address and the base address plus the data length. In this case the write shall not extended beyond the valid range for this source.

Bits 9 to 31 shall be reserved for future use. The value of these bits shall be ignored when read. This field shall be read-only and constant.

7.4.1.2.5 Base Memory Address

The Base Memory Address field shall contain the lowest 32 bits of the full 40-bit RMAP address that shall be used when issuing write commands on this data sink. The address shall use the full 32 bits of the field, where bit 0 is the least significant.

7.4.1.2.6 Data Length Field

The Data Length field shall indicate the length of data accepted by this data sink i.e. the length of individual write operations that should be directed at this target. The Data Length field shall be encoded as shown in Figure 7-17.

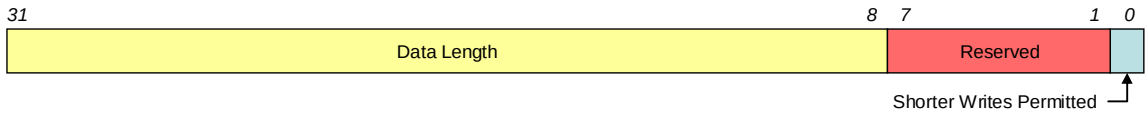


Figure 7-17: Data Length Field Encoding

The Data Length field has two parts:

- Bits 8 to 31 (where 8 is the least significant) shall specify the maximum number of bytes that can and will ever be accepted by this sink.
- Bit 0 shall specify whether this sink permits write operations specifying lengths shorter than the maximum length. If this bit is set, write operations of lengths shorter than the maximum length are permitted; if this bit is set all write operations must be of the maximum length.

Bits 1 to 7 shall be reserved for future use. The value of these bits shall be ignored when read. The Data Length field shall be read-only.

7.4.1.2.7 Lease Timeout Field

The Lease Timeout field shall indicate the duration of the current lease on this sink. The full 32 bits of the field (where 0 is the least significant) shall be used to specify a numeric value in microseconds. The value 0xFFFFFFFF shall be treated distinctly as infinity. Devices shall support the full range of timeouts. The value in this field shall not count down with the lease timer but shall remain constant unless the lease timer expires (reaches zero), upon which the value in this field shall be set to zero by the sink. The value of zero shall indicate that the current lease has expired and the sink is available. If there are queued read commands when the lease expires, the queue shall be flushed and replies shall not be issued to the pending writes. Whenever a write command is received by this target, the lease timer shall be reset to the last value that was written.

7.4.1.2.8 Reply Timeout Field

The Reply Timeout field shall specify the maximum amount of time that shall be allowed to elapse before this sink must issue a reply to a queued write. The full 32 bits of the field (where 0 is the least significant) shall be used to specify a numeric value in microseconds. The value 0xFFFFFFFF shall be treated distinctly as infinity. Devices shall support a single contiguous range of timeouts (not including 0xFFFFFFFF); however, devices may choose not to support the full range. If a value higher than the maximum is written, the field shall be set to the maximum value; if a value lower than the minimum is written, the field shall be set to the minimum value.

7.4.1.3 Initiator Sinks

Initiator Sinks is a complex parameter providing access to zero or more data sink initiators.

Fields in the root entry shall be as shown in Table 7-18.

Table 7-18: Initiator Sinks Parameter Root Entry Fields

ID	Name	Summary
0	Initiator Sink Count	Count of available initiator data sinks
1-255	*Reserved*	*Reserved for future use*

Each initiator is represented by a non-root entry and sinks data of a single, identified type for one target concurrently. Concurrent access is controlled via a timed leasing mechanism. Fields in non-root entries shall be as shown in Table 7-19. There shall be as many non-root entries as there are target sinks identified by the Initiator Sink Count field. The first valid entry shall be entry 1 and valid entries shall be contiguous.

Table 7-19: Initiator Sinks Parameter Non-Root Entry Fields		
ID	**Name**	**Summary**
0	Type/Capabilities/Status	Type of data sunk/sink capabilities and status
1	Destination Address 1	First 4 bytes of destination address
2	Destination Address 2	Middle 4 bytes of destination address
3	Destination Address 3	Last 4 bytes of destination address
4	Command	RMAP command details to use
5	Source Address 1	First 4 bytes of source address
6	Source Address 2	Middle 4 bytes of source address
7	Source Address 3	Last 4 bytes of source address
8	Transaction	RMAP transaction details to use
9	Base Memory Address	Memory address to specify
10	Data Length	Maximum size of data read by this initiator
11	Lease Timer	Length of time remaining on the lease of this sink
12	Reply Timeout	Length of reply timeout on this sink
13-31	*Reserved*	*Reserved for future use*

7.4.1.3.1 Initiator Sink Count Field

The Initiator Sink Count field specifies the number of valid entries in the Initiator Sink parameter. The count shall be represented as an unsigned number from 0 to 255 in bits 0 to 7 with the least significant bit being bit 0. The field encoding is shown in Figure 7-18. Bits 8 to 31 are reserved for future use. The value of these bits shall be ignored. This field shall be read-only and constant.

Figure 7-18: Initiator Sink Count Field Encoding

7.4.1.3.2 Type/Capabilities/Status Field

The Type/ Capabilities/Status field shall define the type of data accepted by this sink, the number of outstanding transactions this sink may have and the current status of the sink. The Type/Capabilities/Status field shall be encoded as shown in Figure 7-19.

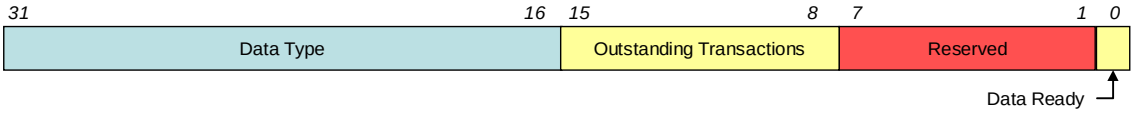


Figure 7-19 Type/ Status Field Encoding

The field shall comprise the following parts:

- Bits 16 to 31 (where 16 is the least significant) shall identify the type of the data accepted by this data sink. Valid types shall be those specified in Table 7-9.
- Bits 7 to 15 (where 7 is the least significant) shall identify the number of outstanding transactions (those awaiting acknowledgement) that this sink may have.
- Bit 0 shall indicate the data availability of this data sink. If the sink is ready to accept data immediately, this bit shall be set to 1; if the sink cannot accept data immediately this bit shall be set to 0.

Bits 1 to 8 shall be reserved for future use. The value of these bits shall be ignored when read. The Type/Capabilities/Status field shall be read-only.

Valid types shall be those specified in Table 7-9 (see Section 7.3.1.2.2). A data type of zero is reserved for future use and shall not be used. Data types from 1 to 255 are allocated and assigned by the SpaceWire Working Group and shall be used in accordance with the released list of data type identifiers and with the relevant data format specifications. The use of data types from 256-65535 shall be vendor specific.

7.4.1.3.3 Destination Address Fields

The Destination Address fields (1-3) encode the destination path address for read operations initiated by this sink. If a path address is not required, and a single logical address is sufficient, all three of these fields shall be zero. The default value of these fields shall be also zero.

Each field shall be split into four 1-byte parts, as shown in Figure 7-20. Path addresses shall be packed into these bytes by placing the last byte of the path address into byte 4 of Destination Address field 3, the second to last byte into byte 3 and so on. All unused bytes in all Destination Address fields should be set to 0.

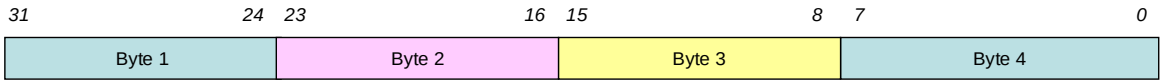


Figure 7-20: Destination Address Field Encoding

Table 7-12 (see section 7.3.1.3.3) shows examples of how path addresses shall be encoded into the Destination Address fields.

7.4.1.3.4 Command Field

The Command Field shall contain the destination logical address, the source logical address, the destination key as well as configuration of the RMAP command to be generated. The Command field shall be encoded as shown in Figure 7-21.

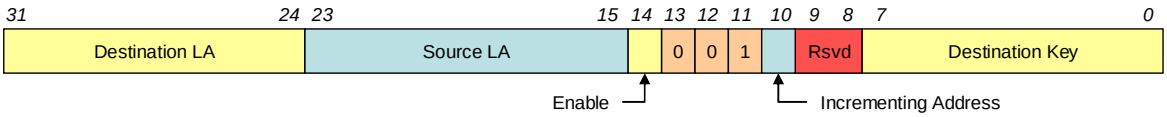


Figure 7-21: Command Field Encoding

The Command field shall have the following parts:

- Bits 24 to 31 (where 24 is the least significant) shall contain the destination logical address to be used by the RMAP read command.
- Bits 15 to 23 (where 15 is the least significant) shall contain the source logical address to be used by the RMAP read command.
- Bit 14 shall enable the initiator associated with this data sink. When this bit is set to 1, the initiator shall generate RMAP read commands requesting sink data. When this bit is set to 0, no commands shall be initiated.
- Bit 13 shall always be 0.
- Bit 12 always be 0.
- Bit 11 shall always be 1.
- Bit 10 shall indicate whether an RMAP read should be generated requesting incrementing addressing. If this bit is set to 1, the initiated RMAP read shall request incrementing addressing; if this bit is set to 0, the initiated RMAP read shall request non-incrementing addressing.
- Bits 0 to 7 (where 0 is the least significant) shall contain the destination key that shall be used by RMAP reads initiated by this sink.

Bits 8 and 9 shall be reserved for future use. The value of these bits shall be ignored when read and shall be written as zeros.

7.4.1.3.5 Source Address Fields

The Source Address fields shall encode the source path address (the return address) in an identical manner to the encoding used for the Destination Address fields (see Section 7.4.1.3.3). If a source path address is not required, these fields shall be set to zero. The default value for the Source Address fields shall be zero.

7.4.1.3.6 Transaction Field

The Transaction field shall contain the highest 8 bits of the transaction identifier and the highest 8 bits of the 40-bit base address for the RMAP read commands initiated by this source. The Transaction field shall be encoded as shown in Figure 7-22.

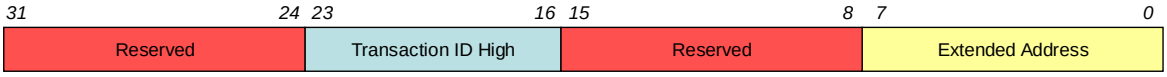


Figure 7-22: Transaction Field Encoding

The Transaction field shall have three parts:

- Bits 15 to 23 (where 15 is the least significant) shall contain the highest 8 bits of the transaction identifier to be used by the RMAP read command. The entry number for this sink shall be used as the lowest 8 bits of the transaction identifier.
- Bits 0 to 7 (where 0 is the least significant) shall contain the extended address to be used by the RMAP read command. This is the top 8 bits of the full 40-bit base memory address.

Bits 8 to 15 and 24 to 31 shall be reserved; these bits shall be ignored when read and shall be written as zero.

7.4.1.3.7 Base Memory Address Field

The Base Memory Address field shall contain the lowest 32 bits of the full 40-bit base memory address to be used by the RMAP read command. Bit 0 shall be the least significant bit.

7.4.1.3.8 Data Length Field

The Data Length field shall indicate the length of data accepted by this data sourced i.e. the length of individual read operations that will be initiated by this source. The Data Length field shall be encoded as shown in Figure 7-23.

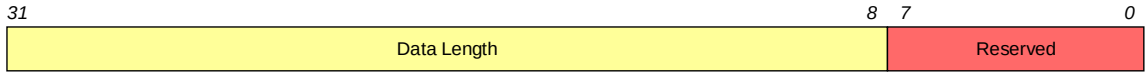


Figure 7-23: Data Length Field Encoding

Bits 8 to 31 (where 8 is the least significant) shall specify the number of bytes that will be read by this sink. Bits 0 to 7 shall be reserved for future use. The value of these bits shall be ignored when read. The Data Length field shall be constant and read-only.

7.4.1.3.9 Lease Timer Field

The Lease Timer field shall indicate the time left before the current owner's lease of this sink expires. The full 32 bits of the field (where 0 is the least significant) shall be used to specify a numeric value in microseconds. The value 0xFFFFFFFF shall be treated distinctly as infinity. Devices shall support the full range of timeouts. The value of zero shall indicate that the current lease has expired and the sink is available. If the initiator is enabled when the lease expires, it shall be disabled. The lease timer begins when a value is written to the Lease Timer field. If a read reply is received, the lease timer shall be reset to the last written value. The lowest 8 bits of the transaction identifier shall be used to associate the reply with a sink entry number.

7.4.1.3.10 Reply Timeout Field

The Reply Timeout field shall specify the maximum amount of time that shall be allowed to elapse before this sink must issue a reply to an initiated read which has not yet received a reply. The full 32 bits of the field (where 0 is the least significant) shall be used to specify a numeric value in microseconds. The value 0xFFFFFFFF shall be treated distinctly as infinity. Devices shall support a single contiguous range of timeouts (not including 0xFFFFFFFF), all devices shall support infinite timeouts; however, devices may choose not to support the full range of timeouts discounting infinity. If a value higher than the maximum is written, the field shall be set to the maximum value; if a value lower than the minimum is written, the field shall be set to the minimum value.

7.4.2 Target Parameter Provision

Device targets shall provide parameters in accordance with the rules shown in Table 7-20. "M" indicates that provision of the parameter/field for the identified operation is mandatory; "O" indicates that provision of the parameter/field for the identified operation is optional; "F" indicates that provision of the parameter/field for the identified operation is forbidden.

Table 7-20: Data Sink Capability Service Target Parameter Provision				
Parameter	**Field(s)**	**Read**	**Write**	**RMW**
Service Information	*All fields*	M*	F	F
Target Sinks	*Root entry fields*	M*	F	F
Target Sinks	Type/Capabilities/Status	M	F	F
Target Sinks	Command, Data Length, Extended Address/Mode, Base Memory Addresss	M*	F	F
Target Sinks	Lease Timeout, Reply Timeout	M	M	O
Initiator Sinks	*Root entry fields*	M*	F	F
Initiator Sinks	Type/Capabilities/Status	M	F	F
Initiator Sinks	*All other fields*	M	M	O

* All field values shall be constant.

7.4.3 Target Actions

The Data Sink capability service provides configuration for a data sink which provides data via RMAP. The target actions specified in this section define the interaction between the SpaceWire-PnP capability service and the RMAP data sink, acting either in the capacity of an RMAP target or initiator or both.

Data Sink Service targets shall implement the Target Available for Data action. Data Sink initiators shall implement the Initiator Available for Data action.

7.4.3.1 Target Available for Data

The Target Available for Data action comprises the following algorithms:

When an RMAP write command is received by the RMAP target, the following algorithm shall be used:

1. If the queue length is zero, an acknowledgement (if required) shall be issued immediately.
    a. If the sink is not available to receive data then the acknowledgement shall not contain data and shall specify the error “RMAP Command not implemented or not authorised”. Data shall not be written to the sink.
    b. If the sink is available to receive data then the acknowledgement (if required) shall indicate success.

2. If the queue length is non-zero the acknowledgement (If required) will be delayed (unless the queue is full).

    a. If the queue is full an acknowledgement (if required) shall be issued immediately specifying the error “RMAP Command not implemented or not authorised”. Data shall not be written to the sink.

    b. If the queue is empty, and the sink is available to accept available, an acknowledgement (if required) shall be issued immediately indicating success and data shall be written to the sink.

    c. If the queue is not full and the sink is not available, the write request shall be queued pending availability and the reply timeout (if configured) shall be started.

When the sink becomes available the following algorithm shall be used:

1. If there is a pending write in the queue then data shall be written to the sink and an acknowledgement (if required) shall be issued.

When the reply timeout expires (becomes zero) the following algorithm shall be used:

1. If there is a pending write in the queue then an acknowledgement shall be issued (if required) specifying the error “RMAP Command not implemented or not authorised”. The reply timeout shall then be restarted.

2. If there is no pending write then the reply timeout shall be allowed to remain expired (at zero).

When the lease timeout expires (becomes zero) the following algorithm shall be used:

1. Any queued writes shall be flushed and no acknowledgements shall be generated.

7.4.3.2 Initiator Available for Data

The Initiator Available for Data action comprises the following algorithms:

When the sink becomes available to accept data the following algorithm shall be used:

1. If the lease timeout is non-zero and the initiator has the capacity for additional outstanding transactions then a read command shall be initiated as specified by the various fields of the relevant Initiator Sinks parameter entry. The operations shall be held as pending reply. The reply timeout shall be started.

2. If the lease timeout is zero or the initiator does not have the capacity for additional outstanding transactions then a read command shall not be initiated.

When a reply is received which matches the transaction ID of the last command to be sent, or the reply timeout field is written to, the following algorithm shall be used:

1. If there is an operation held as pending acknowledgement and the reply is valid and contains data then the received data shall be written to the sink.

When the reply timeout expires (reaches zero) then the following algorithm shall be used:

1. If there is a command held as pending acknowledgement then the read command shall be re-initiated as specified by the various fields of the relevant Initiator Sinks parameter entry. The command shall continue to be held as pending acknowledgement. The reply timeout shall be restarted.
2. If there is no command held as pending then the reply timeout shall be allowed to remain expired (at zero).

When the lease timeout expires (reaches zero) then the following algorithm shall be used:

1. If there is a command held as pending acknowledgement then the command shall be discarded.

7.4.3.3 Target Actions

The Data Sink service shall permit read operations only on all Data sink parameters and write operations where permitted.

7.4.3.3.1 Target Data Sink Operation

Target data sinks have their configuration and status exposed using SpaceWire-PnP whilst the write operation on the actual data sink shall be an RMAP (not SpaceWire-PnP) write command.

If the Reply Timeout field of a target data sink is zero, target data sinks shall reply to write commands on the Write Data field immediately. If the sink is ready for data, the reply shall indicate success. If the sink is not ready for data, the reply shall contain the error “RMAP Command not implemented or not authorised”. If the Reply Timeout field is non-zero, and providing the sink has a queue length of at least 1 (which it must do to support reply timeouts), the reply to a write command on the data sink shall be delayed until either a) data has been consumed by the sink, or b) the reply timeout has elapsed. If the sink has a queue length greater than 1, further write commands on the data sink shall be queued until the first write command has been replied to, until the queue is full. If the queue is full, write commands on the data sink shall be issued with a reply containing the error “RMAP Command not implemented or not authorised”. When a reply is sent to the first write command in the queue, either due to sink availability, or due to reply timeout, the next write command in the queue shall move to the front of the queue and the reply timer shall be restarted. In this manner, the reply timeout shall be measured against the time a command has spent at the front of the queue. If there are queued write commands and the Lease Timer reaches zero (expires), the queue shall be flushed and replies shall not be issued to the pending writes.

7.4.3.3.2 Initiator Data Source Operation

Initiator sources shall initiate RMAP (not SpaceWire-PnP) read commands whenever enabled and the sink is ready to accept data. If an initiator data sink is enabled and the Lease Timer reaches zero (expires), the sink shall be disabled. The Lease Timer shall be reset to the last written value whenever a read reply is received by the sink. The lowest 8 bits of the transaction identifier shall be used to associate both the command and the reply with a source entry number.

7.4.3.4 Initiator Actions

The Data Sink service shall provide all initiators with read access (and write access where permitted) to all Data sink parameters using the Owner Proxy service only, unless the initiator is the owner of the device, in which case access shall be direct.

To access a target data sink, the 40-bit memory address shall be determined by reading the Extended Address/Mode and Base Memory Address fields for the target data sink of interest. An RMAP write command shall then be directed to the device. The incrementing addressing mode of the command shall be set according to the Incrementing Mode flag in the Extended Address/Mode field. If the device is a node, the RMAP write command shall be directed to the device i.e. not to the configuration port of the device, as is the case for SpaceWire-PnP commands. If the device is a router, the RMAP write command shall be directed to the configuration port.

7.4.4 Initiator Primitives

A summary of Data Sink service initiator primitives is presented below:

- SNK_GET_TARGET_COUNT.request
- SNK_GET_TARGET_COUNT.indication
- SNK_READ_TARGET_CONFIG.request
- SNK_READ_TARGET_CONFIG.indication
- SNK_WRITE_TARGET_CONFIG.request
- SNK_WRITE_TARGET_CONFIG.indication
- SNK_WRITE_TARGET_DATA.request
- SNK_WRITE_TARGET_DATA.indication
- SNK_GET_INITIATOR_COUNT.request
- SNK_GET_INITIATOR_COUNT.indication
- SNK_READ_INITIATOR_CONFIG.request

- SNK_READ_INITIATOR_CONFIG.indication
- SNK_WRITE_INITIATOR_CONFIG.request
- SNK_WRITE_INITIATOR_CONFIG.indication
- SNK_INITIATOR_DATA_REQUEST.indication
- SNK_ACK_INITIATOR.request
- SNK_ACK_INITIATOR.indication

These primitives are considered in more detail in the following sections.

7.4.4.1 SNK_GET_TARGET_COUNT.request

The SNK_GET_TARGET_COUNT.request primitive shall permit an initiator to request the count of target sinks provided by the capability service.

The parameters for this primitive shall be:

- RMAP_Parameters
- Capability_Index

RMAP_Parameters includes the source address, destination address and transaction ID.

Capability_Index the numerical index of the data sink capability service in the list provided by the DIDS_READ_CAPABILITY_LIST.indication primitive (see Section 5.2.5.10).

7.4.4.2 SNK_GET_TARGET_COUNT.indication

The SNK_GET_TARGET_COUNT.indication primitive shall provide an indication to an initiator that a target data sink count may be available in response to a SNK_GET_TARGET_COUNT.request. This primitive shall indicate whether the request was successful or not, and may contain a data sink target count.

The parameters for this primitive shall be:

- Result
- Target_Count

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Target_Count parameter may be invalid.

Target_Count shall contain the count of target sinks provided by this capability service (see Section 7.3.1.2.1).

7.4.4.3 SNK_READ_TARGET_CONFIG.request

The SNK_READ_TARGET_CONFIG.request primitive shall permit an initiator to request the current configuration of a data sink target.

The parameters for this primitive shall be:

- RMAP_Parameters
- Capability_Index
- Target_Index

RMAP_Parameters includes the source address, destination address and transaction ID.

Capability_Index the numerical index of the data sink capability service in the list provided by the DIDS_READ_CAPABILITY_LIST.indication primitive (see Section 5.2.5.10).

Target_Index shall indicate the numerical index of the data sink target provided by the capability service.

7.4.4.4 SNK_READ_TARGET_CONFIG.indication

The SNK_READ_TARGET_CONFIG.indication primitive shall provide an indication to an initiator that configuration information for a target data sink may be available in response to a SNK_READ_TARGET_CONFIG.request. This primitive shall indicate whether the request was successful or not, and may contain data sink target configuration.

The parameters for this primitive shall be:

- Result
- Data_Type
- Queue_Length
- Queue_Full
- Data_Ready
- Address_Alignment
- Address_Width
- Destination_LA
- Destination_Key
- Use_Verified

- Use_Acknowledged
- Use_Incrementing
- Extended_Address
- Base_Address
- Offset_OK
- Data_Length
- Shorter_OK
- Lease_Timeout
- Reply_Timeout

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the all other parameters may be invalid.

Data_Type shall indicate the type of data provided by this data sink (see Section 7.4.1.2.2).

Queue_Length shall indicate the maximum length of the data sink queue (see Section 7.4.1.2.2).

Queue_Full shall indicate whether or not the queue is currently full (see Section 7.4.1.2.2).

Data_Ready shall indicate whether or not the sink is currently ready for data (see Section 7.4.1.2.2).

Address_Alignment shall indicate the addressing alignment required by the sink (see Section 7.4.1.2.2).

Address_Width shall indicate the addressing width used by the sink (see Section 7.4.1.2.2).

Destination_LA shall indicate the destination logical address value to use when accessing the sink, or zero if any logical address can be used (see Section 7.4.1.2.3).

Destination_Key shall indicate the destination key value to use when accessing the sink (see Section 7.4.1.2.3).

Use_Verified shall indicate whether or not the writes addressed to the data sink should use verified writes (see Section 7.4.1.2.3).

Use_Acknowledged shall indicate whether or not the the writes addressed to the data sink should use acknowledged writes (see Section 7.4.1.2.3).

Use_Incrementing shall indicate whether or not writes addressed to the data sink should use incrementing addressing mode (see Section 7.4.1.2.3).

Extended_Address shall indicate the extended address to use when accessing the sink (see Section 7.4.1.2.4).

Base_Address shall indicate the base address to use when accessing the sink (see Section 7.4.1.2.5).

Offset_OK shall indicate whether or not the target accepts writes which are offset from the extended address/base address specified by the Extended_Address and Base_Address parameters (see Section 7.4.1.2.5).

Data_Length shall indicate the length of data provided by the sink (see Section 7.4.1.2.6).

Shorter_OK shall indicate whether or not the target accepts writes which are shorter than the length specified by the Data_length parameter (see Section 7.4.1.2.6).

Lease_Timeout shall indicate the length of the current lease timeout, or zero if the sink is not currently leased (see Section 7.4.1.2.7).

Reply_Timeout shall indicate the length of the current reply timeout, or zero if no reply timeout is required (see Section 7.4.1.2.8).

7.4.4.5 SNK_WRITE_TARGET_CONFIG.request

The SNK_WRITE_TARGET_CONFIG.request primitive shall permit an initiator to set the current configuration of a data sink target.

The parameters for this primitive shall be:

- RMAP_Parameters
- Capability_Index
- Target_Index
- Lease_Timeout
- Reply_Timeout

RMAP_Parameters includes the source address, destination address and transaction ID.

Capability_Index the numerical index of the data sink capability service in the list provided by the DIDS_READ_CAPABILITY_LIST.indication primitive (see Section 5.2.5.10).

Target_Index the numerical index of the data sink target provided by the capability service.

Lease_Timeout shall indicate the length of the desired lease timeout, or zero if the sink is not to be leased (see Section 7.4.1.2.7).

Reply_Timeout shall indicate the length of the current reply timeout, or zero if no reply timeout is required (see Section 7.4.1.2.8).

7.4.4.6 SNK_WRITE_TARGET_CONFIG.indication

The SNK_WRITE_TARGET_CONFIG.indication primitive shall provide an indication to an initiator that configuration information for a target data sink may have been written in response to a SNK_WRITE_TARGET_CONFIG.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

7.4.4.7 SNK_WRITE_TARGET_DATA.request

The SNK_READ_TARGET_DATA.request primitive shall permit an initiator to issue an RMAP write to transmit data to a target data source, if it is available.

The parameters for this primitive shall be:

- RMAP_Parameters
- Destination_Key
- Verified
- Acknowledged
- Incrementing
- Extended_Address
- Base_Address
- Data
- Data_Length

RMAP_Parameters includes the source address, destination address and transaction ID.

Destination_Key shall indicate the destination key to use for the RMAP write operation.

Verify shall indicate whether or not the RMAP write operation should use a verified write.

Acknowledge shall indicate whether or not the RMAP write operation should use an acknowledged write.

Incrementing shall indicate whether or not the RMAP read operation should use an incrementing write.

Extended_Address shall indicate the extended address to use for the RMAP write operation.

Base_Address shall indicate the base address to use for the RMAP write operation.

Data shall contain the data to write to the sink.

Data_Length shall indicate the length of data present in the Data parameter.

7.4.4.8 SNK_WRITE_TARGET_DATA.indication

The SNK_READ_TARGET_DATA.indication primitive shall provide an indication to an initiator the success, or otherwise of a SNK_WRITE_TARGET_DATA.request primitive. If the SNK_WRITE_TARGET_DATA.request primitive requested an acknowledged write then this primitive will indicate the successful receipt, or otherwise, of the acknowledgement.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

7.4.4.9 SNK_GET_INITIATOR_COUNT.request

The SNK_GET_INITIATOR_COUNT.request primitive shall permit an initiator to request the count of initiator sinks provided by the capability service.

The parameters for this primitive shall be:

- RMAP_Parameters
- Capability_Index

RMAP_Parameters includes the source address, destination address and transaction ID.

Capability_Index the numerical index of the data sink capability service in the list provided by the DIDS_READ_CAPABILITY_LIST.indication primitive (see Section 5.2.5.10).

7.4.4.10 SNK_GET_INITIATOR_COUNT.indication

The SNK_GET_INITIATOR_COUNT.indication primitive shall provide an indication to an initiator that an initiator data sink count may be available in response to a SNK_GET_INITIATOR_COUNT.request. This primitive shall indicate whether the request was successful or not, and may contain a data sink initiator count.

The parameters for this primitive shall be:

- Result
- Initiator_Count

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the Target_Count parameter may be invalid.

Initiator_Count shall contain the count of initiator sinks provided by this capability service (see Section 7.3.1.3.1).

7.4.4.11 SNK_READ_INITIATOR_CONFIG.request

The SNK_READ_INITIATOR_CONFIG.request primitive shall permit an initiator to request the current configuration of a data sink initiator.

The parameters for this primitive shall be:

- RMAP_Parameters
- Capability_Index
- Initiator_Index

RMAP_Parameters includes the source address, destination address and transaction ID.

Capability_Index the numerical index of the data sink capability service in the list provided by the DIDS_READ_CAPABILITY_LIST.indication primitive (see Section 5.2.5.10).

Initiator_Index shall indicate the numerical index of the data sink initiator provided by the capability service.

7.4.4.12 SNK_READ_INITIATOR_CONFIG.indication

The SNK_READ_INITIATOR_CONFIG.indication primitive shall provide an indication to an initiator that configuration information for an initiator data sink may be available in response to a SNK_READ_INITIATOR_CONFIG.request. This primitive shall indicate whether the request was successful or not, and may contain data sink initiator configuration.

The parameters for this primitive shall be:

- Result
- Data_Type
- Outstanding_Transactions

- Data_Ready
- Enabled
- Destination_Path
- Destination_LA
- Destination_Key
- Use_Incrementing
- Source_Path
- Source_LA
- Transaction_ID
- Extended_Address
- Base_Address
- Data_Length
- Lease_Timeout
- Reply_Timeout

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported. If Result does not indicate success then the all other parameters may be invalid.

Data_Type shall indicate the type of data provided by this data sink (see Section 7.4.1.3.2).

Outstanding_Transactions shall indicate the maximum number of initiated reads awaiting replies which may be handled by the data sink (see Section 7.4.1.3.2).

Data_Ready shall indicate whether or the sink is available to accept data (see Section 7.4.1.3.2).

Enabled shall indicate whether or not the initiator sink is currently enabled (see Section 7.4.1.3.2).

Destination_Path shall contain the path address portion (i.e. excluding the final logical address) of the destination address; this may be empty (see Section 7.4.1.3.3).

Destination_LA shall contain the logical address of the destination (see Section 7.4.1.3.4).

Destination_Key shall indicate the destination key value to use when initiating reads from the sink (see Section 7.4.1.3.4).

Use_Incrementing shall indicate whether or not the sink initiator should initiate incrementing reads (see Section 7.4.1.3.4).

Source_Path shall contain the path address portion (i.e. excluding the final logical address) of the source address; this may be empty (see Section 7.4.1.3.5).

Source_LA shall contain the logical address of the source (see Section 7.4.1.3.4).

Transaction_ID shall contain the upper byte of the transaction ID used by reads initiated from this sink (see Section 7.4.1.3.6).

Extended_Address shall indicate the extended address used by reads initiated from this sink (see Section 7.4.1.3.6).

Base_Address shall indicate the base address used by reads initiated from this sink (see Section 7.4.1.3.7).

Data_Length shall indicate the length of data to be read by reads initiated from this sink (see Section 7.4.1.3.8)

Lease_Timeout shall indicate the length of the current lease timeout, or zero if the sink is not currently leased (see Section 7.4.1.3.9).

Reply_Timeout shall indicate the length of the current reply timeout, or zero if no reply timeout is in use (see Section 7.4.1.3.10).

7.4.4.13 SNK_WRITE_INITIATOR_CONFIG.request

The SNK_WRITE_INITIATOR_CONFIG.request primitive shall permit an initiator to set the current configuration of a data source initiator.

The parameters for this primitive shall be:

- RMAP_Parameters
- Capability_Index
- Target_Index
- Enabled
- Destination_Path
- Destination_LA
- Destination_Key
- Use_Incrementing

- Source_Path
- Source_LA
- Transaction_ID
- Extended_Address
- Base_Address
- Lease_Timeout

RMAP_Parameters includes the source address, destination address and transaction ID.

Capability_Index the numerical index of the data sink capability service in the list provided by the DIDS_READ_CAPABILITY_LIST.indication primitive (see Section 5.2.5.10).

Target_Index the numerical index of the data sink initiator provided by the capability service.

Enabled shall indicate whether or not the initiator sink should be enabled (see Section 7.4.1.3.2).

Destination_Path shall contain the path address portion (i.e. excluding the final logical address) of the destination address; this may be empty (see Section 7.4.1.3.3).

Destination_LA shall contain the logical address of the destination (see Section 7.4.1.3.4).

Destination_Key shall indicate the destination key value to use when initiating reads from the sink (see Section 7.4.1.3.4).

Use_Incrementing shall indicate whether or not the sink initiator should initiate incrementing reads (see Section 7.4.1.3.4).

Source_Path shall contain the path address portion (i.e. excluding the final logical address) of the source address; this may be empty (see Section 7.4.1.3.5).

Source_LA shall contain the logical address of the source (see Section 7.4.1.3.4).

Transaction_ID shall contain the upper byte of the transaction ID to be used by reads initiated from this sink (see Section 7.4.1.3.6).

Extended_Address shall indicate the extended address to be used by reads initiated from this sink (see Section 7.4.1.3.6).

Base_Address shall indicate the base address to be used by reads initiated from this sink (see Section 7.4.1.3.7).

Data_Length shall indicate the length of data to be read by reads initiated from this sink (see Section 7.4.1.3.8)

Lease_Timeout shall indicate the length of the desired lease timeout, or zero if the sink is not to be leased (see Section 7.4.1.3.9).

Reply_Timeout shall indicate the length of the desired reply timeout, or zero if no reply timeout is required (see Section 7.4.1.3.10).

7.4.4.14 SNK_WRITE_INITIATOR_CONFIG.indication

The SNK_WRITE_INITIATOR_CONFIG.indication primitive shall provide an indication to an initiator that configuration information for an initiator data sink may have been written in response to a SNK_WRITE_INITIATOR_CONFIG.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

7.4.4.15 SNK_INITIATOR_DATA_REQUEST.indication

The SNK_INITIATOR_DATA_REQUEST.indication primitive shall provide an indication to an initiator that data is requested due to the receipt of an initiated read from an initiator data sink.

The parameters for this primitive shall be:

- Transaction_ID
- Data_Length

Transaction_ID shall indicate the full 16-bit transaction ID of the received read.

Data_Length shall indicate the length of data required by the received read.

7.4.4.16 SNK_ACK_INITIATOR.request

The SNK_ACK_INITIATOR.request primitive shall permit an initiator to request the transmission of a read reply in response to the receipt of read command (indicated by a SNK_INITIATOR_DATA_REQUEST.indication primitive).

The parameters for this primitive shall be:

- RMAP_Parameters
- Status
- Data

- Data_Length

RMAP_Parameters includes the source address, destination address and transaction ID.

Status shall indicate the RMAP status code to include in the acknowledgement.

Data shall contain the data to be transmitted as part of the read reply.

Data_Length shall indicate the length of data available in the Data parameter.

7.4.4.17 SNK_ACK_INITIATOR.indication

The SNK_ACK_INITIATOR.indication primitive shall provide an indication to an initiator that a read reply to an initiator data sink may have been sent in response to a SNK_ACK_INITIATOR.request. This primitive shall indicate whether the request was successful or not.

The parameters for this primitive shall be:

- Result

Result shall indicate the success, or otherwise, of this primitive. Result shall support all RMAP error codes; additional error codes may be supported.

7.4.5 Initiator Primitive Provision

Initiators shall provide primitives in accordance with the rules shown in Table 7-14. “M” indicates that provision of the primitive is mandatory; “O” indicates that provision of the primitive is optional. Where an optional request primitive is provided, an indication primitive shall also be provided, and vice-versa.

Table 7-21: Data Source Capability Service Initiator Primitive Provision		
Primitive	**Provision**	**Notes**
SNK_GET_TARGET_COUNT.request	M	-
SNK_GET_TARGET_COUNT.indication	M	-
SNK_READ_TARGET_CONFIG.request	M	-
SNK_READ_TARGET_CONFIG.indication	M	-
SNK_WRITE_TARGET_CONFIG.request	M	-
SNK_WRITE_TARGET_CONFIG.indication	M	-
SNK_WRITE_TARGET_DATA.request	M	-
SNK_WRITE_TARGET_DATA.indication	M	-
SNK_GET_INITIATOR_COUNT.request	M	-
SNK_GET_INITIATOR_COUNT.indication	M	-
SNK_READ_INITIATOR_CONFIG.request	M	-
SNK_READ_INITIATOR_CONFIG.indication	M	-
SNK_WRITE_INITIATOR_CONFIG.request	M	-
SNK_WRITE_INITIATOR_CONFIG.indication	M	-
SNK_INITIATOR_DATA_REQUEST.indication	M	-
SNK_ACK_INITIATOR.request	M	-
SNK_ACK_INITIATOR.indication	M	-

8 IMPLEMENTATION USING RMAP

This normative section defines the usage of RMAP by SpaceWire-PnP.

8.1.1 Targets and Initiators

All network devices shall behave as an SpaceWire-PnP target: receiving requests and responding to them appropriately. Active nodes shall behave as SpaceWire-PnP initiators. Additionally, all devices may behave as RMAP targets and/or initiators to permit support of data source and sink capability services.

8.1.2 SpaceWire Addressing

RMAP defines support for using path addressing and logical addressing. SpaceWire PnP devices shall support both path addressing and logical addressing.

8.1.3 Operations

RMAP provides three types of operation: Read, Write and Read-Modify-Write. The usage of these operations in SpaceWire-PnP shall be as follows.

8.1.3.1 Read

All SpaceWire-PnP devices shall support read operations as a SpaceWire-PnP target.

8.1.3.2 Write

The RMAP write operation has four variants:

- Unacknowledged, unverified write
- Acknowledge, unverified write
- Unacknowledged, verified write
- Acknowledge, verified write

All SpaceWire-PnP devices shall support acknowledged, verified writes only.

8.1.3.3 Read-Modify-Write (RMW)

All level 2 SpaceWire-PnP devices shall support a specific implementation of the RMW command. RMW operations shall always be 32-bit (4-byte) operations. The RMW operation shall be implemented as a conditional write, as follows:

1. the SpaceWire PnP device shall read the address specified;

2. the read value shall be compared to the mask value contained in the RMW packet;
3. if the read value and the mask value match, the data value shall be written to the same address;
4. the read value shall be returned to the initiator of the command in a reply packet.

The data value and mask value fields are as defined in the RMAP standard.

Level 1 devices may support RMW commands, if the command is supported it shall be as described above.

8.1.4 Addressing Mode and Width

Incrementing address mode shall always be used. Each address location shall be 32-bits (4 bytes) wide.

8.1.5 Data Byte-Ordering (Endianness)

The order of the four bytes making up each 32-bit field in the data cargo shall be big-endian, i.e. most significant byte first. For example, if the hexadecimal number 0x12345678 were to be packed into the data cargo, the following bytes would be transmitted, in order: 0x12 0x34 0x56 0x78.

8.1.6 Field Usage

RMAP packets follow a regular format with consistent field meanings across read, write and RMW commands. The following sections detail the usage of these fields for SpaceWire-PnP.

8.1.6.1 Destination Address

SpaceWire-PnP packets shall always be addressed to the configuration port of SpaceWire-PnP devices as described in Section 3.2.1. A logical address of 0xFE shall be used.

8.1.6.2 Protocol Identifier

The protocol identifier field should carry the protocol identifier for SpaceWire-PnP. Note that this is different to the RMAP protocol identifier.

8.1.6.3 Key

For commands addressed to devices (both level 1 and level 2), the key shall be set to 32 (0x20). For commands addressed to owner proxy address spaces (level 2 networks only), the key shall be set to the proxy key value.

8.1.6.4 Transaction Identifier

The transaction identifier field contains a transaction identifier assigned by the initiator. Any value of transaction identifier may be used. The SpaceWire-PnP device shall include the same transaction identifier in a corresponding reply, as defined by the RMAP standard.

8.1.6.5 Extended Address

For commands addressed to devices (both level 1 and level 2), the extended address shall be set to zero. For commands addressed to owner proxy address spaces (level 2 networks only), the extended address shall be set to the proxy ID.

8.1.6.6 Address Encoding

The RMAP address field shall be encoded as shown in Figure 8-1.

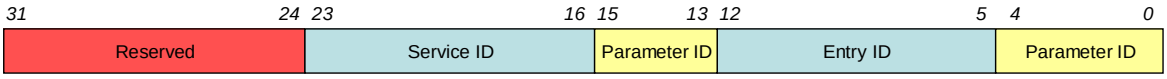


Figure 8-1: Address Field Encoding

The 32-bit address field shall be split into 4 parts:

- Bits 16 to 23 (where 16 is the least significant) shall contain the service ID.
- Bits 13 to 15 (where 13 is the least significant) shall contain the parameter ID.
- Bits 5 to 12 (where 5 is the least significant) shall contain the entry ID for complex parameters, or zero when accessing simple parameters.
- Bits 0 to 4 (where 0 is the least significant) shall contain the field ID.

Valid values for each of these fields are detailed in Sections 5 and 6. Bits 24 to 31 shall be reserved for future use and shall be set to zero by initiators and ignored by targets.

A SpaceWire-PnP device receiving a request which results in an access containing invalid values for these fields shall reply with the error 'RMAP Command not implemented or not authorised'. A write command spanning multiple fields, the first of which are valid, may result in those fields being modified.

8.1.6.7 Data Length

The data length field shall always be a multiple of four bytes.

8.1.6.8 Reply Error Codes

SpaceWire PnP devices shall use errors codes as specified in the RMAP standard.

9 SUPPORT FOR THE SPW-10X

The SpW-10X routing device (Atmel part number AT7910E) has all the features required of a level SpaceWire-PnP device. The protocol used for configuration is RMAP; the usage of RMAP and the address space are defined in the SpW-10X User Manual [RD4]. This section defines how a SpW-10X device should be integrated into a SpaceWire-PnP network.

9.1 DETECTION

An active node may attempt to detect a SpW-10X only under the following conditions:

1. If a link is indicated as active.
2. No SpaceWire-PnP read reply was received within the applicable timeout.

A SpW-10X shall be detected by reading the Device Manufacturer and Chip ID register and verifying the Manufacturer Code, Chip ID and Version Code against those specified in the User Manual [RD4].

9.2 SERVICE PROVISION

The SpW-10X has sufficient features to be able to support the functionality of all mandatory services of SpaceWire-PnP. A full set of initiator primitives shall therefore be supported for the SpW-10X. It is the responsibility of an active node implementation to provide primitives which correctly set the SpW-10X registers.

9.2.1 Device Identification Service

Initiator primitives shall operate as though the SpW-10X implemented the Device Identification service parameters as documented in this section.

The following parameters shall be treated as not implemented:

- Vendor String
- Product String
- Capability List

Fields in the Device Information parameter shall map to constant values and SpW-10X registers as shown in Table 9-1.

Table 9-1: Device Information Parameter Field Mapping for the SpW-10X		
ID	**Name**	**Mapping**
0	Vendor ID/ Product ID	Contains UoD vendor ID and product ID 0x0001
1	Region/Number of Ports	Contains 0x0000008A
2	Static Device ID High	Contains 0x00000000
3	Static Device ID Low	Contains 0x00000000
4	Version/Instance ID	Contains version code from SpW-10X Device Manufacturer and Chip ID Register (8 bits) in the top 8 bits of this field; all other bits shall be zero.
5	Operation/String Lengths	Contains 0x01010000

Fields in the Device Status parameter shall map to constant values and SpW-10X registers as shown in Table 9-2.

Table 9-2: Device Status Parameter Field Mapping for the SpW-10X		
ID	**Name**	**Mapping**
0	Network ID	Mirrors the contents of the Router Identity Register from the SpW-10X
1	Active Ports	Bits 1 to 10 shall contain bits 8 to 17 of the Network Discovery Register on the SpW-10X; all other bits shall be 0
2	Return Port/Operational Status	Bits 8 to 11 shall contain bits 4 to 7 of the Network Discovery Register on the SpW-10X; all other bits shall be 0

9.2.2 Network Management Service

Initiator primitives shall operate as though the SpW-10X implemented the Network Management service parameters as documented in this section.

Fields in the Network Identification parameter shall map to constant values and SpW-10X registers as shown in Table 9-3.

Table 9-3: Network Identification Parameter Field Mapping for the SpW-10X		
ID	**Name**	**Mapping**
0	Network ID	Mirrors the contents of the Router Identity Register from the SpW-10X
1	Device Logical Address	Contains the constant value 0x00000000

9.2.3 Link Configuration Service

Initiator primitives shall operate as though the SpW-10X implemented the Link Configuration service parameters as documented in this section.

Fields in the Port Activity parameter shall map to constant values and SpW-10X registers as shown in Table 9-4.

Table 9-4: Port Activity Parameter Field Mapping for the SpW-10X		
ID	**Name**	**Mapping**
0	Active Ports	Bits 1 to 10 shall contain bits 8 to 17 of the Network Discovery Register on the SpW-10X; all other bits shall be 0
1	Active Ports Delta	Contains the constant value 0xFFFFFFFF

Fields in the Reference Rate parameter shall map to constant values and SpW-10X registers as shown in Table 9-5.

Table 9-5: Reference Rate Parameter Field Mapping for the SpW-10X		
ID	**Name**	**Mapping**
0	Base Rate/Mode	Bits 16 to 31 shall be the constant value 0x1000. Bits 0 to 15 shall be set to correctly represent the maximum frequency of the SpW-10X, this can be determined by reading bits 16 to 20 of the Transmit Clock Control register (see [RD4]) and translating accordingly assuming the input clock frequency is known. If this is not known then a 30MHz input clock should be assumed.
1	Maximum/ Minimum Reference Rate	Contains the constant value 0x00080002
2	Reference Rate	This shall be mapped to bits 0 to 1 of the Transmit Control Register on the SpW-10X with appropriate translation from literal integer values for the Reference Rate parameter to codes for the SpW-10X register transmit divider (see [RD4]).

The Link Count field in the root entry of the Link Control parameter shall contain the constant value 0x0000000A.

There shall be 10 non-root entries in the Link Control parameter (one for each port), the values in each field shall depend on the entry number (i.e. the port number) and shall be mapped as indicated in Table 9-6.

Table 9-6: Link Control Parameter Non-Root Entry Field Mapping for the SpW-10X		
ID	**Name**	**Mapping**
0	Link Type/Status	Bit 31 shall indicate the port type, this shall be set to 1 for entries (ports) 1 to 8 and set to 0 for entries (ports) 9 and 10. Bits 30 shall be set to 1. Bits 8 to 29 shall be set to 0. Bit 0 and bits 2 to 7 shall be set according to the status bits in the relevant Port Control/Status register of the SpW-10X (see [RD4]). A write of zero to this field shall clear all errors by an appropriate write to the SpW-10X Error Active register (see [RD4]).
1	Maximum/Minimum Link Rate	Shall contain appropriate values depending on the setting of the Reference Rate and the port type (SpaceWire port or external port). The range specified here shall not include invalid values, such as those that would generate a transmit rate of less than 2Mbps (see guidance and lookup table in [RD4]).
2	Link Rate/Priority/State	Bits 16 to 31 shall contain the link rate divider determined from bits 16 to 22 of the Port Control/Status register of the SpW-10X. Initiator primitives shall prevent invalid values being written, this is both values out of the range of the Maximum/Minimum link rate, and value which would result in an invalid initiatialisation data rate or invald duty cycle should not be permitted (see formulae and lookup table in [RD4]). Bits 8 to 15 shall be 0x00. Bit 7 shall correspond to the relevant bit (this depends on the entry/port number) in the Time-Code Enable register of the SpW-10X. Bits 0 and 1 shall represent and control the link state via appropriate manipulation of the link state in the Port Control/Status register.

9.2.4 Router Configuration Service

Initiator primitives shall operate as though the SpW-10X implemented the Router Configuration service parameters as documented in this section.

Fields in the Router Control parameter shall map to constant values and SpW-10X registers as shown in Table 9-7.

Table 9-7: Router Control Parameter Field Mappings for he SpW-10X

ID	Name	Mapping
0	Watchdog Timeout	This field shall represent the SpW-10X watchdog timeout value as specified by bits 0 to 3 of the Router Control register on the SpW-10X (see RD4]). When written to, the closest watchdog timer value shall be chosen as specified in Section 5.5.2.1.1 and this value shall be selected appropriately in the SpW-10X Router Control register.
1	Arbitration Mode	Contains the constant value 0x00000006
2	Time-Code Counters	The SpW-10X recognises all time-codes and records the top two bits. To represent this in the Time-Code counters field, the current time-code value (from the Time-Code register on the SpW-10X) shall be populated in the channel corresponding to the top two bits. All other time-code counters shall be set to zero. All channel enable bits shall contain the same value; if any bit (and consequently all bits) is set to 0, this shall clear the Time-Code Enable register in the SpW-10X; if any bits (and consequently all bits) is set to 1 then the Time-Code Enable register in the SPW-10X shall be populated as specified in Section 9.2.3. The reset bits shall not be supported.

The SpW-10X supports a complete routing table of 223 valid logical addresses; the Logical Address Count field in the root entry of the Routing Table parameter shall therefore contain the constant value 0x000000DF.

Fields in the non-root entries of the Routing Table parameter shall map to constant values and SpW-10X registers as shown in Table 9-8.

Table 9-8: Routing Table Parameter Non-Root Entry Field Mappings for the SpW-10X

ID	Name	Mapping
0	Port Association	Bits 1 to 10 of this field shall map directly to bits 1 to 10 of the appropriate GAR Table register on the SpW-10X according to the entry number.
1	Address Control	Bits 9 to 15 shall be set to 0. Bit 8 shall map directly onto bit 30 of the appropriate GAR Table register on the SpW-10X according to the entry number. Bits 2 and 3 shall be set to 0. Bit 1 shall map directly onto bit 29 of the appropriate GAR Table register on the SpW-10X according to the entry number. Bit 0 shall map onto the inverse of bit 31 of the appropriate GAR Table register on the SpW-10X according to the entry number.

9.3 Unrepresented SpW-10X Configuration Options

There are a number of SpW-10X configuration options which are not represented by SpaceWire-PnP services as they implement features beyond those identified in the SpaceWire standard. These configuration options shall be left as the power on defaults (see [RD4]).

9.4 Level 2 Network Integration

For the purposes of level 2 network integration, the SpW-10X, with associated service/parameter mappings as described in Section 9.2, shall be treated as a SpaceWire-PnP level 1 device. The SpW-10X implementation of RMAP RMW shall not be used.

10 DOCUMENT CHANGES

The changes made this document are listed in this section.

10.1 VERSION 1.1

Table 10-1: Changes to Document to Version 1.1	
Section/Reference	**Change**
All sections	Typographical errors and minor inconsistencies resolved.
Section 5.2.2	Datasheet Library parameter removed; functionality to be provided via a data source which would be used without leasing.
Section 7.1	Data source target now supplies data via RMAP, not SpaceWire-PnP.
Section 7.4	Data sink target now supplies data via RMAP, not SpaceWire-PnP.

10.2 VERSION 2.0

This document underwent a major restructuring, introducing level 1 and 2 services and reducing the complexity of the addressing scheme. The section changes are therefore too extensive to enumerate here.

10.3 VERSION 2.1

Previous versions of the document left Section 9 (Support for the SpW-10X) unwritten (marked *TBW*: To Be Written). A skeletal set of mappings has been introduced in Version 2.1.