Logical Domains Agents Specification

Revision 1.2

2/11/2010

contact: alexandre.chartre@sun.com

Revision	Date	Author	Comments
1.0	08/18/2009	Alexandre Chartre	Initial revision
1.1	11/03/2009	Jim Marks	Add message type to get pci device info
1.2	02/11/2010	Michael Christensen	Add chassis serial number message to 'agent-system'

Table of Contents

1. Introduction

This document describes the structure of messages used by Logical Domains Agents. These messages are sent between the control domain and LDoms agents using the domain services framework.

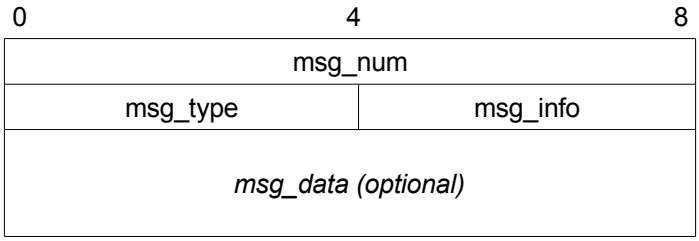
2. Versions

The structure of messages is defined by the version of the agent. An agent can support several versions and uses the version mechanism provided by the domain service framework. In that framework, a version is defined as a major and a minor number (*major.minor*).

- the major number defines the common structure of messages supported by the agent. This document defines the common structure of messages for a major number of 1.
- the minor number is a version number private to the agent. It is usually used to track changes in the structure of messages of the agent.

3. Message Common Structure

Any message sent or received by an agent with a major version number of 1 should have the following structure:



Unless specified otherwise, any size and offset in this document is specified as a number of bytes.

offset	size	field name	description
0	8	msg_num	message number
8	4	msg_type	message type
12	4	msg_info	message information
16	variable	msg_data	message data (optional)

If an agent receives a message whose size is smaller than the minimum size required to have a correct message structure (16 bytes) then the agent should ignore the message.

- **message number (msg_num):** unless specified otherwise for an agent, this field is an identifier of the message and it contains the following information:
  - if the message is a request then the field contains a monotonically increasing number provided by the requestor to uniquely identify the request.
  - if the message is a reply as a result of a particular request then the field contains the value provided by the requestor to identify the original request. This allows the requestor to match a reply with the corresponding original request.
- **message type (msg_type):** this field contains a value indicating the type of the message. Each agent can

implement its own set of message types using any value from the interval [0x1, 0x7FFF]. Values 0x0 and [0x8000, 0x8FFF] are reserved for defining generic message types which can be used by any agent. Refer to section 4 (*Generic Messages*) for the list of generic messages.

- **message info (msg_info):** this field contains additional information about the message. The interpretation of this field depends on the message type and its usage by the agent. For example, this field can be used to indicate the size of data present in the message data (msg_data) field.
- **message data (msg_data):** the field contains additional data about the message. The interpretation and the size of the field depends on the message type and its usage by the agent. This field is optional and can be empty. When this field is not empty, its size should be a multiple of 8 bytes.

4. Generic Messages

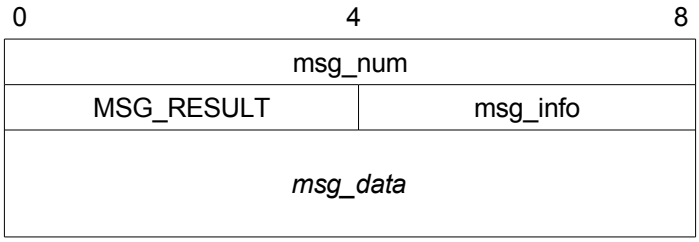
The following message types (msg_type) are defined as generic message types which can be used by any agent with a major version number of 1:

Name	Value	Description
MSG_RESULT	0x8000	result message
MSG_ERROR	0x8001	error message

An agent can send one of these messages as a reply message after receiving a message from the control domain. Such a reply message has to be sent to the domain from which the original message was received. If an agent can not send a reply message (for example, because the other domain is down) then the reply message and the original message should be dropped.

4.1. Result (MSG_RESULT)

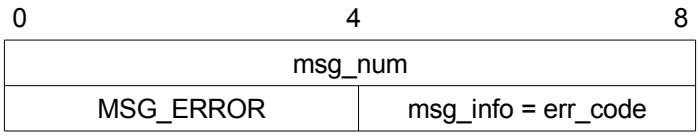
A result message has the following structure:



An agent shall send a result message to indicate that a message has been received and processed, and to return the result of processing. The msg_info and msg_data fields can be used to provide the result of the processing.

4.2. Error (MSG_ERROR)

An error message has the following structure:



An agent shall send an error message to indicate that the agent has not been able to process a message. In that

case, the msg_info field is interpreted as err_code and used to describe the error that has occurred.

offset	size	field name	description
12	4	err_code	error code

- **error code (err_code):** code describing the error. This can be one of the following values:

Name	Value	Description
MSGERR_FAIL	0x0000	message has failed
MSGERR_INVALID	0x8000	message is invalid
MSGERR_NOTSUP	0x8001	message is not supported
MSGERR_DENY	0x8002	message is denied

  - **MSGERR_FAIL:** generic failure code.
  - **MSGERR_INVALID:** indicate that the message received has an invalid structure or it contains an invalid request, or a request with invalid arguments.
  - **MSGERR_NOTSUP:** indicate that the message received has a message type which is not supported by the agent.
  - **MSGERR_DENY:** indicate that the processing of the message has been denied. An agent should reply with an MSG_ERROR message with err_code set to MSGERR_DENY whenever it receives a message from a domain which is not the control domain.

  Error code values 0x0 and [0x8000, 0x8FFF] are reserved for defining generic error codes which can be used by any agent. An agent can also add its own err_code values to describe an error. In that case, the error code should be set with a value from the interval [0x1, 0x7FFF]. The interpretation of the value is then specific to each agent.

5. Agent Specific Messages

For each agent, the following sections describe message types that can be defined for a specific agent. Unless specified otherwise, when receiving a message, an agent will reply with a result message (MSG_RESULT) if it was able to process the request, or with an error message (MSG_ERROR) if the processing was not completed or not possible.

An agent is described with the following information:

- **Name:** the name of the agent. This name is used as the service identifier to register the agent as a domain service.
- **Version:** the version of the agent.
- **Description:** a short description of the purpose of the agent.
- **Message types:** the list of message types defined for the agent.

In addition, each message type is described with the following information:

- **Description:** the purpose of this message.
- **Content:** the content of the message.
- **Reply:** the reply message sent by the agent after receiving and processing (or not) the message.

6. Device Agent

- **Name:** agent-device
- **Version:** 1.0
- **Description:** the device agent provides information about devices in the domain the agent is running on.
- **Message Types:**

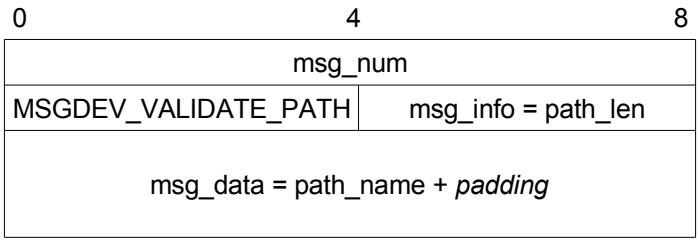
Name	Value	Description
MSGDEV_VALIDATE_PATH	0x01	validate a path name
MSGDEV_VALIDATE_NIC	0x02	validate a network interface

6.1. Validate Path (MSGDEV_VALIDATE_PATH)

Description:

This message is a request to validate a path name inside a domain.

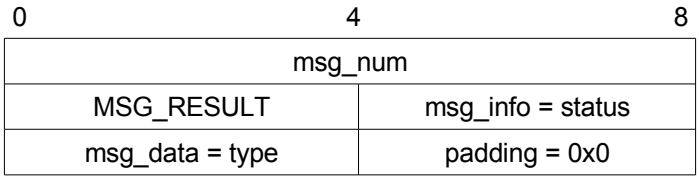
Content:



- *msg_num:* message number as described in section 4.
- *msg_type:* MSGDEV_VALIDATE_PATH
- *msg_info:* length in bytes of the path name available in the msg_data field.
- *msg_data:* the path name to validate (for example /dev/dsk/c0t11d0s2), it can be followed by some padding so that the size of the msg_data field is a multiple of 8 bytes. The path name does not need to be terminated by a null character; the number of bytes constituting the path name is indicated in the msg_info field.

Reply:

- A MSGREQ_RESULT message with the following structure is sent after the processing of the request has been completed:



The status field indicates the result of the validation of the path name. It contains a bitwise combination of the following values:

Name	Value	Description
DEVPATH_EXIST	0x01	path exists and is accessible
DEVPATH_OPENRW	0x02	path can be opened read/write
DEVPATH_OPENRO	0x04	path can be opened read-only

In addition, if the path name exists (i.e. DEVPATH_EXIST is set in the status field), then the type field contains a 32-bit value describing the type of file the path name is pointing to. This value can be:

Name	Value	Description
DEVPATH_TYPE_UNKNOWN	0x00	the path points to an unknown type
DEVPATH_TYPE_FILE	0x01	the path points to a regular file
DEVPATH_TYPE_DEVICE	0x02	the path points to a device

If the path does not exist or the type of file can not be identified then the type field should be set to 0x00.

- A MSGREQ_ERROR is sent if the processing of the request was not completed or not possible.

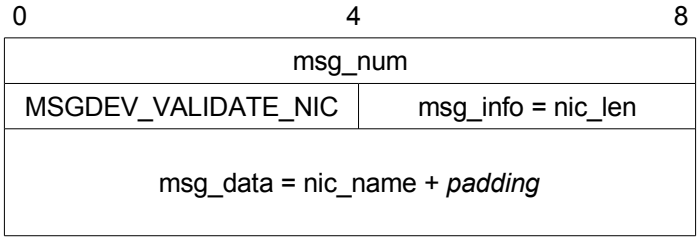
The message is terminated by a padding value of 0x0.

6.2. Validate NIC (MSGDEV_VALIDATE_NIC)

Description:

This message is a request to validate a network interface inside a domain.

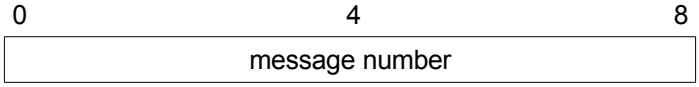
Content:



- *msg_num:* message number as described in section 4.
- *msg_type:* MSGDEV_VALIDATE_NIC
- *msg_info:* length in bytes of the network interface name available in the msg_data field.
- *msg_data:* the name of the network interface to validate (for example nxge0), it can be followed by some padding so that the size of the msg_data field is a multiple of 8 bytes. The network interface name does not need to be terminated by a null character; the number of bytes constituting the network interface name is indicated in the msg_info field.

Reply:

- A MSGREQ_RESULT message with the following structure is sent after the processing of the request has been completed:



MSG_RESULT	msg_info = status

The status field indicates the result of the validation of the network interface name. It contains a bitwise combination of the following values:

Name	Value	Description
DEVNIC_EXIST	0x01	the network interface exists

The msg_data field is not used.

- A MSGREQ_ERROR is sent if the processing of the request was not completed or not possible.

7. System Agent

- **Name:** agent-system
- **Version:** 1.0
- **Description:** the system agent provides information about the system in the domain the agent is running on.
- **Message Types:**

Name	Value	Description
MSGSYS_GET_SYSINFO	0x01	get system information
MSGSYS_GET_CHASSISNO	0x02	Get chassis number

7.1. Get System Information (MSGSYS_GET_SYSINFO)

Description:

This message is a request to get system information.

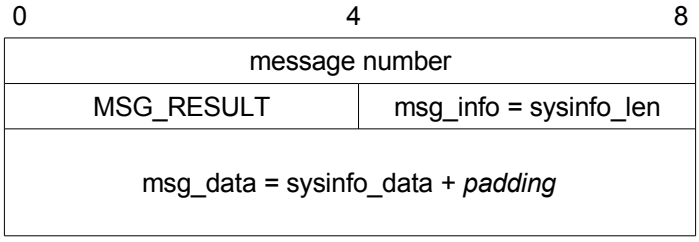
Content:

0	4	8
msg_num		
MSGSYS_GET_SYSINFO	msg_info = 0x0	

- *msg_num:* message number as described in section 4.
- *msg_type:* MSGSYS_GET_SYSINFO
- *msg_info:* this field is not used and should be set to 0.
- *msg_data:* this field is not used and should be empty.

Reply:

- A MSGREQ_RESULT message with the following structure is sent after the processing of the request has been completed:



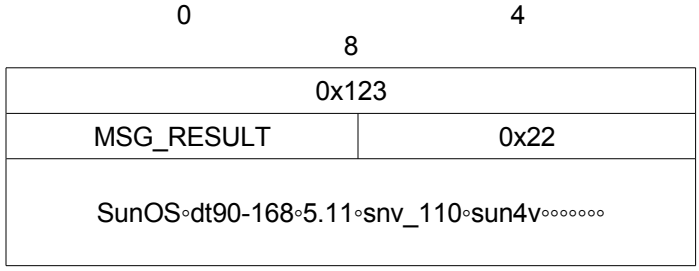
Then the msg_info contains the length in bytes of the sysinfo_data available in the msg_data field. The sysinfo_data contains six strings containing the following information in this order:

- operating system name
- network name of the system
- release level of the operating system
- version level of the operation system
- hardware type of the system

Each string is separated from the following string by a one byte value of 0x0. The last string is followed by a one byte value of 0x0. A string can be empty if the domain is unable to provide the corresponding information; in that case, the information appears as a single one byte value of 0x0 (i.e. the empty string followed by a one byte value of 0x0). The sysinfo_data information can be followed by some padding values so that the size of the msg_data field is a multiple of 8 bytes.

Implementation guideline: if the domain is running an operating system that supports the uname(2) system call (such as Solaris or Linux) then the six information strings can be extracted from the utsname structure returned by the uname(2) system call.

Example: the following message is an example of a result message that an agent can send after having successfully processed a MSGSYS_GET_SYSINFO message with the message number 0x123:



In this example, the symbol ◦ represents the value 0x0. The sysinfo_data information is followed by 6 0x0 values so that the total size of msg_info is a multiple of 8 bytes.

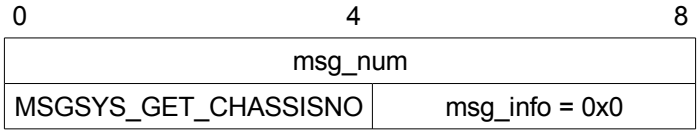
- A MSGREQ_ERROR is sent if the processing of the request was not completed or not possible.

7.2. Get Chassis Number (MSGSYS_GET_CHASSISNO)

Description:

This message is a request to get the platform chassis serial number from the control domain.

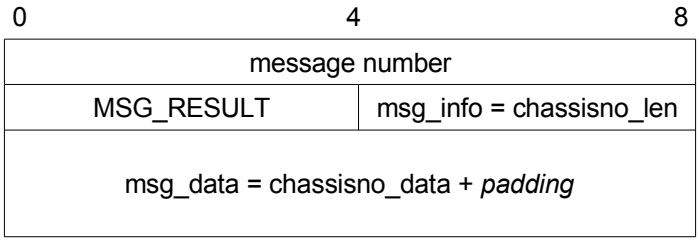
Content:



- *msg_num:* message number as described in section 4.
- *msg_type:* MSGSYS_GET_CHASSISNO
- *msg_info:* this field is not used and should be set to 0.
- *msg_data:* this field is not used and should be empty.

Reply:

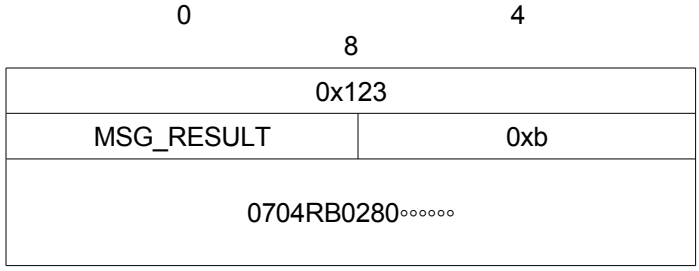
- A MSGREQ_RESULT message with the following structure is sent after the processing of the request has been completed:



Then the msg_info contains the length in bytes of the chassisno_data available in the msg_data field. The chassisno_data contains a single string indicating the platform's chassis serial number. This string is terminated by a one byte value of 0x0. The chassisno_data information can be followed by some padding values so that the size of the msg_data file is a multiple of 8 bytes.

Implementation guideline: on sun4v systems, the chassis serial number is available only within the PRI MD of the control domain. A request for a chassis serial number on a non-control domain will result in an MSG_FAIL error message.

Example: the following message is an example of a result message that an agent can send after having successfully processed a MSGSYS_GET_CHASSISNO message with the message number 0x123:



In this example, the symbol ◦ represents the value 0x0. The sysinfo_data information is followed by 6 0x0 values so that the total size of msg_info is a multiple of 8 bytes.

- A MSGREQ_ERROR is sent if the processing of the request was not completed or not possible. Since the chassis serial number is only available on the control domain, a request to a non-control domain will result in a MSG_FAIL error.

8. Direct IO (DIO) Agent

- **Name:** agent-dio
- **Version:** 1.0
- **Description:** The dio agent provides information about the pcie devices on the current system.
- **Message Types:**

Name	Value	Description
MSGDIO_PCIDEV_INFO	0x1	Get PCI device information

8.1. Get PCI Device Information (MSGDIO_PCIDEV_INFO)

Description:

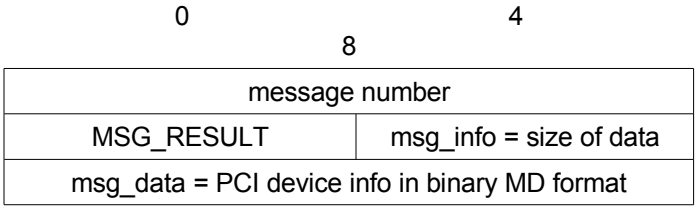
This message is a request to get PCI device information.

Content:

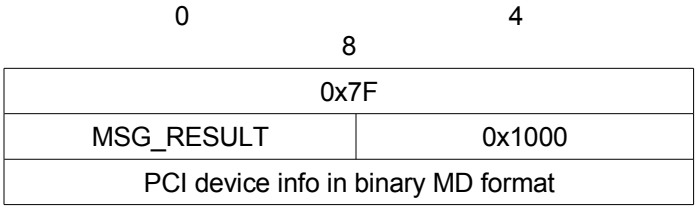
- *msg_num*: message number as described in section 4.
- *msg_type*: MSGDIO_PCIDEV_INFO
- *msg_info*: this field is not used and should be set to 0.
- *msg_data*: this field is not used and should be empty.

Reply:

. A MSGREQ_RESULT message with the following structure is sent after the processing of the request has been completed:



Example: the following message is an example of a result message that the dio agent can send after having successfully processed an MSGDIO_PCIDEV_INFO message with the message number 0x7F and containing a data element of 0x1000 bytes. The binary MD format in the msg_data field is specified in the md-transport.pdf file of the following case: http://sac.sfbay/FWARC/2005/115/



The msg_info field contains the length in bytes of the Machine Description (MD) contained in the msg_data field. The MD itself consists of a tree of device nodes which represent the PCI devices on the target system. Each node of the MD contains data (properties) describing a single PCI device. Each node will be of type “iodevice” and contain the following properties, each of which has the same semantics as the corresponding property in the prom device tree:

Property Name	Tag	Required	Description
dev_path	PROP_STR	yes	A string representing the device path of the corresponding device in Solaris device path format
device_type	PROP_STR	yes	Same as the prom device tree node (string)
compatible	PROP_DATA	yes	An array of string names for this node.
device_id	PROP_VAL	yes	64-bit unsigned integer repesenting PCI Device ID
vendor_id	PROP_VAL	yes	64-bit unsigned integer representing PCI Vendor ID
class_code	PROP_VAL	yes	64-bit unsinged integer representing PCI Device Class ID

. A MSGREQ_ERROR is sent if the processing of the request was not completed or not possible.