ODI Changes from Revision 1 to Revision 2

This document summarizes the changes between from Revision 1 to Revision 2 of the ODI Preliminary Specifications.

The changes are driven by three key factors: The adoption of VITA 49.2, the documentation of a standard API, to be named ODI-A, and the general review and testing of the ODI specifications

VITA 49.2

VITA 49.2, abbreviated as V49.2, supersedes the original V49.0 specification. It is a pure superset of capability, adding control and other functionality. The original ODI specifications referenced V49.0. All references are changed to V49.2. In most cases, V49.2 retained backwards compatibility, but there are a few exceptions, which are noted in this document. Figures changed, so the updated ODI specifications reference V49.2 figures and text.

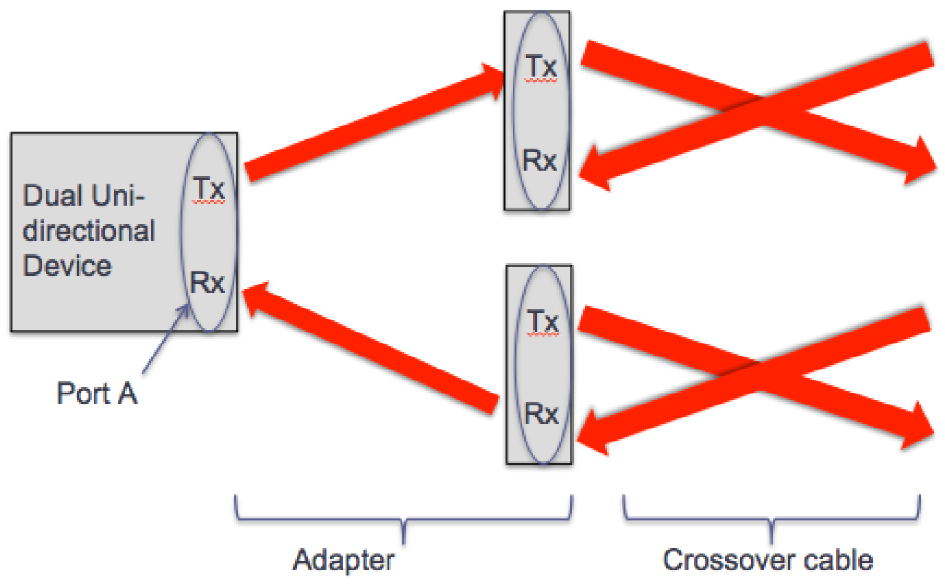
ODI-1:

ODI-1 documents the physical and protocol layers of the Optical Data Interface. ODI-1 does not specify the packet format, so is largely unaffected by the change to V49.2. However, ODI-1 shares a common glossary with the other standards, and occasionally makes an observation about another standard. For this reason, there are slight changes to ODI-1.

There are terminology changes throughout the specifications due to the change to V49.2. For example, IF Data Packets are renamed Signal Data Packets in V49.2.

Dual uni-directional devices: ODI-1 adds explicit permission to configure a port as a dual uni-directional device. That is, the receivers can receive data from one device, and transmit it to another from the same port. This may be useful for a device that is a signal processor, for example. The recommended method to connect these devices is to have an adapter that breaks the original port into two ports, one receiving and one transmitting, to be connected to standard ODI cables.

Optical Cable Adapter for dual uni-directional devices



The figure above shows how an adapter could split a port into a receiving port and a transmitting port.

ODI has also updated the state diagrams and details for single port streaming, with and without flow control.

ODI-2:

ODI-2 is the first ODI specification to refer to V49, and is impacted by the change to V49.2. The key mandatory change for data packets is that bit 25 of the Signal Data Packet header is changed from “0” to “1”. This indicates that it is no longer a V49.0 packet, but a V49.2 (or higher) packet. This change is also implemented for Context Packets and the newly defined Command Packets.

Another change is that V49.2 has defined a new packet type, called a Command Packet. Command Packets are useful for sending command information to an arbitrary waveform generator or other exciters, specifically metadata about the signal (e.g. bandwidth, IF offset frequency, etc.). This was previously performed using Context Packets. As in the past, Context Packets and Command Packets remain optional capability. Since Context and Command packets have very similar properties, ODI allows either to control a consumer. However, if a consumer can execute Command or Context packets, it must execute Command packets as the mandatory method. Execution of Context Packets is allowed if the device is explicitly enabled to do so through the control plane. This may be a useful feature for an AWG playing back recorded sample data, as only Context Packets would have been recorded. Only one Command Packet subtype, the Control Packet, is expected to be used in ODI systems.

ODI-2 has also changed the default Timestamp Fields in the Prologue for devices that do not implement Timestamps. TSI/TSF has changed from 11/11 to 11/01. This is due to 11/11 already being a legal combination for other functionality.

It should be noted that ODI-2 has grown in size due to the VITA addition of Command Packets and various additional Context Fields. ODI-2 has documented how ODI devices should implement these if the developer finds them to be necessary in their application. However, many of the packet subtypes are expected never to be used in ODI systems. As with the first revision, only the Signal Data Packet subtype is needed for a fully functioning ODI system.

ODI-2 has added a number of rules for Port Aggregation. The default Stream IDs on aggregated ports now increment by 1024 instead of 64. This allows more automatic assigning of Stream ID of downstream channels. Also, more details have been added for using Flow Control with Port Aggregation.

ODI-2.1

ODI-2.1 defines standard data format for Signal Data Packets. The key change is including the indicator in the header that this is no longer V49.0. ODI-2.1 has added a table of common data formats along with their Class IDs.

Context Packets remain optional, but a standard Context Packet is defined if used. There have been changes to the standard Context Packet to match field locations in the standard Control Packet. Most significant is the elimination of Device ID, and the inclusion of two null fields CIF1 and CIF2.

Command Packets are optional, but a standard Control Packet is defined, that is similar to the standard context packet. The other subtypes of Command Packets are not used.

Context and Command Packets

Context and Command Packets are optional. In V49.2, Context Packets are a method for a producer to report metadata about the signal. Command Packets are a method of sending metadata to a consumer to be executed. In this sense, they are duals of each other. ODI-2.1 adopts the Command Packet structure as the standard method of controlling a consumer, such as a signal generator. ODI-2.1 also allows a consumer to also execute a context packet. This is useful in record/playback applications where the original meta-data was recorded as Context Packets. In order for a device to do either, ODI-2.1 has made the data fields the same for both packet types to the maximum extent possible.

ODI-A

A new standard, ODI-A, has been introduced coincident with Rev. 2 of the ODI Specifications. ODI-A defines the API to be used in test and measurement applications to access and control the ODI functionality of a device.

Besides the API definition, the ODI Specification web site links to helpful API examples. This includes:

- A .chm file that is a Windows help file for an example instrument (model Mxxxx) with an ODI interface. This file was created using the Pacific Mindworks Nimbus software tool.

- A C/C++ include file defining the ODI interface for the Mxxx instrument. This file was created using the Pacific Mindworks Nimbus software tool.

- A C/C++ include file defining common ODI enums and types.