

Compare Results

Old File:

USB_PD_R3_1 V1.8 2023-04_Ch8.pdf

428 pages (11.22 MB)

13/10/2023 19:37:23

versus

New File:

USB_PD_R3_2 V1.0 2023-10_Ch 8.pdf

577 pages (10.99 MB)

31/10/2023 18:10:16

Total Changes

10568

Text only comparison

Content

5163
2889
2516

Replacements
Insertions
Deletions

Styling and Annotations

0 Styling
0 Annotations

[Go to First Change \(page 1\)](#)

8. Device Policy

8.1 Overview

This section describes the Device Policy and Policy Engine that implements it. For an overview of the architecture and how the Device Policy Manager fits into this architecture, please see [Section 2.6 "Architectural Overview"](#).

8.2 Device Policy Manager

The Device Policy Manager is responsible for managing the power used by one or more USB Power Delivery ports. In order to have sufficient knowledge to complete this task it needs relevant information about the device it resides in. Firstly, it has a priori knowledge of the device including the capabilities of the power supply and the receptacles on each Port since these will for example have specific current ratings. It also has to know information from the USB-C Port Control module regarding cable insertion, type and rating of cable etc. It also has to have information from the power supply about changes in its capabilities as well as being able to request power supply changes. With all of this information the Device Policy Manager is able to provide up to date information regarding the capabilities available to a specific Port and to manage the power resources within the device.

When working out the capabilities for a given Source Port the Device Policy Manager will take into account firstly the current rating of the Port's receptacle and whether the inserted cable is PD or non-PD rated and if so, what is the capability of the plug. This will set an upper bound for the capabilities which might be offered. After this the Device Policy Manager will consider the available power supply resources since this will bound which Voltages and currents might be offered. Finally, the Device Policy Manager will consider what power is currently allocated to other ports, which power is in the Power Reserve and any other amendments to Policy from the System Policy Manager. The Device Policy Manager will offer a set of capabilities within the bounds detailed above.

When selecting a capability for a given Sink Port the Device Policy Manager will look at the capabilities offered by the Source. This will set an upper bound for the capabilities which might be requested. The Device Policy Manager will also consider which capabilities are required by the Sink in order to operate. If an appropriate match for Voltage and Current can be found within the limits of the receptacle and cable, then this will be requested from the Source. If an appropriate match cannot be found then a request for an offered Voltage and current will be made, along with an indication of a capability mismatch.

USB PD defines two types of power sources:

- Pre-defined Voltage sources (Fixed, Variable and Battery)
- Programmable Voltage sources:
 - Programmable Power Supply (PPS)
 - Adjustable Voltage Supply (AVS)

The first are generally used for classic charging wherein the charger electronics reside inside the Sink. The Device Policy Manager in the Sink requests a fixed Voltage from the list of PDOs offered by the Source and which is converted internally to charge the Sink's battery and/or power its function.

The second moves the charger electronics that manage the Voltage control outside the Sink and back into the Source itself. When in SPR PPS operation, the Device Policy Manager in the Sink requests a specific Voltage with a 20mV accuracy and sets a current limit. Unlike traditional USB where Sinks are responsible for limiting the current, they consume, the SPR PPS Source limits the current to what the Sink has requested. When operating in AVS mode, the Device Policy Manager in the Sink requests a specific Voltage with a 100mV accuracy and requests a maximum current it is allowed to draw. Note that the AVS Sources unlike SPR PPS Sources do not support current limit mode. A Sink operating in AVS Mode is responsible not to draw more current than it requests.

The process to request power is the same for both types of power Sources although the actual format and contents of the request are slightly different. The primary operational differences are:

- A Sink that is using SPR PPS is required to periodically sent requests to let the Source know it is still alive and communicating. When this communication fails a Hard Reset results.
- A Sink operating in SPR mode has no special timing requirements.
- A Sink operating in EPR mode is required to periodically communicate with the Source to let it know it is still operational. If the communication fails, a Hard Reset results.

For Dual-Role Power Ports the Device Policy Manager manages the functionality of both a Source and a Sink. In addition, it is able to manage the Power Role Swap process between the two. In terms of power management this could mean that a Port which is initially consuming power as a Sink is able to become a power resource as a Source. Conversely, Attached Sources might request that power be provided to them.

The functionality within the Device Policy Manager (and to a certain extent the Policy Engine) is scalable depending on the complexity of the device, including the number of different power supply capabilities and the number of different features supported for example System Policy Manager interface or Capability Mismatch, and the number of ports being managed. Within these parameters it is possible to implement devices from very simple power supplies to more complex power supplies or devices such as USB hubs or Hard Drives. Within multiport devices it is also permitted to have a combination of USB Power Delivery and non-USB Power Delivery ports which **Should** all be managed by the Device Policy Manager.

As noted in [Section 2.6 “Architectural Overview”](#) the logical architecture used in the PD specification will vary depending on the implementation. This means that different implementations of the Device Policy Manager might be relatively small or large depending on the complexity of the device, as indicated above. It is also possible to allocate different responsibilities between the Policy Engine and the Device Policy Manager, which will lead to different types of architectures and interfaces.

The Device Policy Manager is responsible for the following:

- Maintaining the Local Policy for the device.
- For a Source, monitoring the present capabilities and triggering notifications of the change.
- For a Sink, evaluating and responding to capabilities related requests from the Policy Engine for a given Port.
- Control of the Source/Sink in the device.
- Control of the USB-C Port Control module for each Port.
- Interface to the Policy Engine for a given Port.

The Device Policy Manager is responsible for the following **Optional** features when implemented:

- Communications with the System Policy over USB.
- For Sources with multiple ports monitoring and balancing power requirements across these ports.
- Monitoring of batteries and AC power supplies.
- Managing Modes in its Port Partner and Cable Plug(s).

8.2.1 Capabilities

The Device Policy Manager in a Provider **Shall** know the power supplies available in the device and their capabilities. In addition, it **Shall** be aware of any other PD Sources of power such as batteries and AC inputs. The available power sources and existing demands on the device **Shall** be taken into account when presenting capabilities to a Sink.

The Device Policy Manager in a Consumer **Shall** know the requirements of the Sink and use this to evaluate the capabilities offered by a Source. It **Shall** be aware of its own power sources e.g., Batteries or AC supplies where these have a bearing on its operation as a Sink.

The Device Policy Manager in a Dual-Role Power Device **Shall** combine the above capabilities and **Shall** also be able to present the dual-role nature of the device to an Attached PD Capable device.

8.2.2 System Policy

A given PD Capable device might have no USB capability, or PD might have been added to a USB device in such a way that PD is not integrated with USB. In these two cases there **Shall** be no requirement for the Device Policy Manager to interact with the USB interface of the device. The following requirements **Shall** only apply to PD devices that expose PD functionality over USB.

The Device Policy Manager **Shall** communicate over USB with the System Policy Manager according to the requirements detailed in [\[UCSI\]](#). Whenever requested the Device Policy Manager **Shall** implement a Local Policy according to that requested by the System Policy Manager. For example, the System Policy Manager might request that a battery powered Device temporarily stops charging so that there is sufficient power for an HDD to spin up.

Note: that due to timing constraints, a PD Capable device **Shall** be able to respond autonomously to all time-critical PD related requests.

8.2.3 Control of Source/Sink

The Device Policy Manager for a Provider **Shall** manage the power supply for each PD Source Port and **Shall** know at any given time what the negotiated power is. It **Shall** request transitions of the supply and inform the Policy Engine whenever a transition completes.

The Device Policy Manager for a Consumer **Shall** manage the Sink for each PD Sink Port and **Shall** know at any given time what the negotiated power is.

The Device Policy Manager for a Dual-Role Power Device **Shall** manage the transition between Source/Sink roles for each PD Dual-Role Power Port and **Shall** know at any given time what operational role the Port is in.

8.2.4 Cable Detection

8.2.4.1 Device Policy Manager in a Provider

The Device Policy Manager in the Provider **Shall** control the USB-C Port Control module and **Shall** be able to use the USB-C Port Control module to determine the Attachment status.

Note: that it might be necessary for the Device Policy Manager to also initiate additional discovery using the [Discover Identity](#) Command in order to determine the full capabilities of the cabling (see [Section 6.4.4.3.1 "Discover Identity"](#)).

8.2.4.2 Device Policy Manager in a Consumer

The Device Policy Manager in a Consumer controls the USB-C Port Control module and **Shall** be able to use the USB-C Port Control module to determine the Attachment status.

8.2.4.3 Device Policy Manager in a Consumer/Provider

The Device Policy Manager in a Consumer/Provider inherits characteristics of Consumers and Providers and **Shall** control the USB-C Port Control module in order to support the Dead Battery back-powering case to determine the following for a given Port:

- Attachment of a USB Power Delivery Provider/Consumer which supports Dead Battery back-powering.
- Presence of V_{BUS} .

8.2.4.4 Device Policy Manager in a Provider/Consumer

The Device Policy Manager in a Provider/Consumer inherits characteristics of Consumers and Providers and **May** control the USB-C Port Control module in order to support the Dead Battery back-powering case to determine the following for a given Port:

- Presence of V_{BUS} .

8.2.5 Managing Power Requirements

The Device Policy Manager in a Provider **Shall** be aware of the power requirements of all devices connected to its Source Ports. This includes being aware of any reserve power that might be required by devices in the future and ensuring that power is shared optimally amongst Attached PD Capable devices. This is a key function of the Device Policy Manager; whose implementation is critical to ensuring that all PD Capable devices get the power they require in a timely fashion in order to facilitate smooth operation. This is balanced by the fact that the Device Policy Manager is responsible for managing the sources of power that are, by definition, finite.

The Consumer's Device Policy Manager **Shall** ensure that it takes no more power than is required to perform its functions and gives back unneeded power whenever possible (in such cases the Provider **Shall** maintain a Power Reserve to ensure future operation is possible).

8.2.5.1 Managing the Power Reserve

There might be some products where a Device has certain functionality at one power level and a greater functionality at another, for example a Printer/Scanner that operates only as a printer with one power level and as a scanner if it can get more power. Visibility of the linkage between power and functionality will only be apparent at the USB Host; however, the Device Policy Manager provides the mechanisms to manage the power requirements of such Devices.

Devices with the GiveBack flag cleared report Operating Current and Maximum Operating Current (see [Section 6.4.2.2 "GiveBack Flag"](#)). For many Devices the Operating Current and the Maximum Operating Current will be the same. Devices with highly variable loads, such as Hard Disk Drives, might use Maximum Operating Current.

Devices with the GiveBack flag set report Operating Current and Minimum Operating Current (see [Section 6.4.2.2 "GiveBack Flag"](#)). For many Devices the Operating Current and the Minimum Operating Current will be the same. Devices that charge their own batteries might use the Minimum Operating Current and GiveBack flag.

For example, in the first case, a mobile device might require 500mA to operate, but would like an additional 1000mA to charge its Battery. The mobile device would set the GiveBack flag (see [Section 6.4.2.2 "GiveBack Flag"](#)) and request 500mA in the Minimum Operating Current field and 1500mA in the Operating Current field (provided that 1500mA was offered by the Source) indicating to the Provider that it could temporarily recover the 1000mA to meet a transitory request.

In the second case, a Hard Disk Drive (HDD) might require 2A to spin-up, but only 1A to operate. At startup the HDD would request Maximum Operating Current of 2A and an Operating Current of 2A. After the drive is spun-up and ready to operate it would make another request of 1A for its Operating Current and 2A for its Maximum Operating Current. Over time, its inactivity timers might expire, and the HDD will go to a lower power state. When the HDD is next accessed, it has to spin-up again. So, it will request an Operating Current of 2A and a Maximum Operating Current of 2A. The Provider might have the extra power available immediately and can immediately honor the request. If the power is not available, the Provider might have to harvest power, for example use the [GotoMin](#) Message to get back some power before honoring the HDD's request. In such a case, the HDD would be told to wait via a [Wait](#) Message. The HDD continues to Request additional power until the request is finally granted.

It **Shall** be the Device Policy Manager's responsibility to allocate power and maintain a Power Reserve so as not to over-subscribe its available power resource. A Device with multiple ports such as a Hub **Shall** always be able to meet the incremental demands of the Port requiring the highest incremental power from its Power Reserve.

The **GotoMin** Message is designed to allow the Provider to reclaim power from one Port to support a Consumer on another Port that temporarily requires additional power to perform some short-term operation. In the example above, the mobile device that is being charged reduces its charge rate to allow a Device Policy Manager to meet a request from an HDD for start-up current required to spin-up its platters. Any power which is available to be reclaimed using a **GotoMin** Message **May** be counted as part of the Power Reserve.

A Consumer requesting power **Shall** take into account its operational requirements when advertising its ability to temporarily return power. For example, a mobile device with a Dead Battery that is being used to make a call **Should** make a request that retains sufficient power to continue the call. When the Consumer's requirements change, it **Shall** re-negotiate its power to reflect the changed requirements.

8.2.5.2 Power Capability Mismatch

A capability mismatch occurs when a Consumer cannot obtain required power from a Provider (or the Source is not PD Capable) and the Consumer requires such capabilities to operate. Different actions are taken by the Device Policy Manager and the System Policy Manager in this case.

8.2.5.2.1 Local device handling of mismatch

The Consumer's Device Policy Manager **Shall** cause a Message to be displayed to the end user that a power capability mismatch has occurred. Examples of such feedback can include:

- For a simple Device an LED **May** be used to indicate the failure. For example, during connection the LED could be solid amber. If the connection is successful, the LED could change to green. If the connection fails, it could be red or alternately blink amber.
- A more sophisticated Device with a user interface, e.g., a mobile device or monitor, **Should** provide notification through the user interface on the Device.

The Provider's Device Policy Manager **May** cause a Message to be displayed to the user of the power capability mismatch.

Because the capability mismatch **might** not cause operational failure, the Provider's Device Policy Manager **Should Not** display a message to the user if the power offered to the Sink meets or exceeds the Sink Minimum PDP Advertised in the **Sink_Capabilities_Extended** Message (see [Section 6.5.13 "Sink_Capabilities_Extended Message"](#)). If a message is displayed, it **Should Not** be shown as an error unless the power offered to the Sink is less than the Sink Minimum PDP Advertised in the **Sink_Capabilities_Extended** Message.

8.2.5.2.2 Device Policy Manager Communication with System Policy

In a USB Power Delivery aware system with an active System Policy manager (see [Section 8.2.2 "System Policy"](#)), the Device Policy Manager **Shall** notify the System Policy Manager of the mismatch. This information **Shall** be passed back to the System Policy Manager using the mechanisms described in [\[UCSI\]](#). The System Policy Manager **Should** ensure that the user is informed of the condition. When another Port in the system could satisfy the Consumer's power requirements the user **Should** be directed to move the Device to the alternate Port.

In order to identify a more suitable Source Port for the Consumer the System Policy Manager **Shall** communicate with the Device Policy Manager in order to determine the Consumer's requirements. The Device Policy Manager **Shall** use a **Get_Sink_Cap** Message (see [Section 6.3.8 "Get_Sink_Cap Message"](#)) to discover which power levels can be utilized by the Consumer.

8.2.6 Use of “Unconstrained Power” bit with Batteries and AC supplies

The Device Policy Manager in a Provider or Consumer **May** monitor the status of any variable sources of power that could have an impact on its capabilities as a Source such as Batteries and AC supplies and reflect this in the “Unconstrained Power” bit (see [Section 6.4.1.2.2.3 “Unconstrained Power”](#) and [Section 6.4.1.3.1.3 “Unconstrained Power”](#)) provided as part of the Source or Sink Capabilities Message (see [Section 6.4.1 “Capabilities Message”](#)). When monitored, and a USB interface is supported, the External Power status (see [\[UCSI\]](#)) and the Battery state (see [Section 9.4.1 “GetBatteryStatus”](#)) **Shall** also be reported to the System Policy Manager using the USB interface.

8.2.6.1 AC Supplies

The Unconstrained Power bit provided by Sources and Sinks (see [Section 6.4.1.2.2.3 “Unconstrained Power”](#) and [Section 6.4.1.3.1.3 “Unconstrained Power”](#)) notifies a connected device that it is acceptable to use the Advertised power for charging as well as for what is needed for normal operation. A device that sets the Unconstrained Power bit has either an external source of power that is sufficient to adequately power the system while charging external devices or expects to charge external devices as a primary state of function (such as a battery pack).

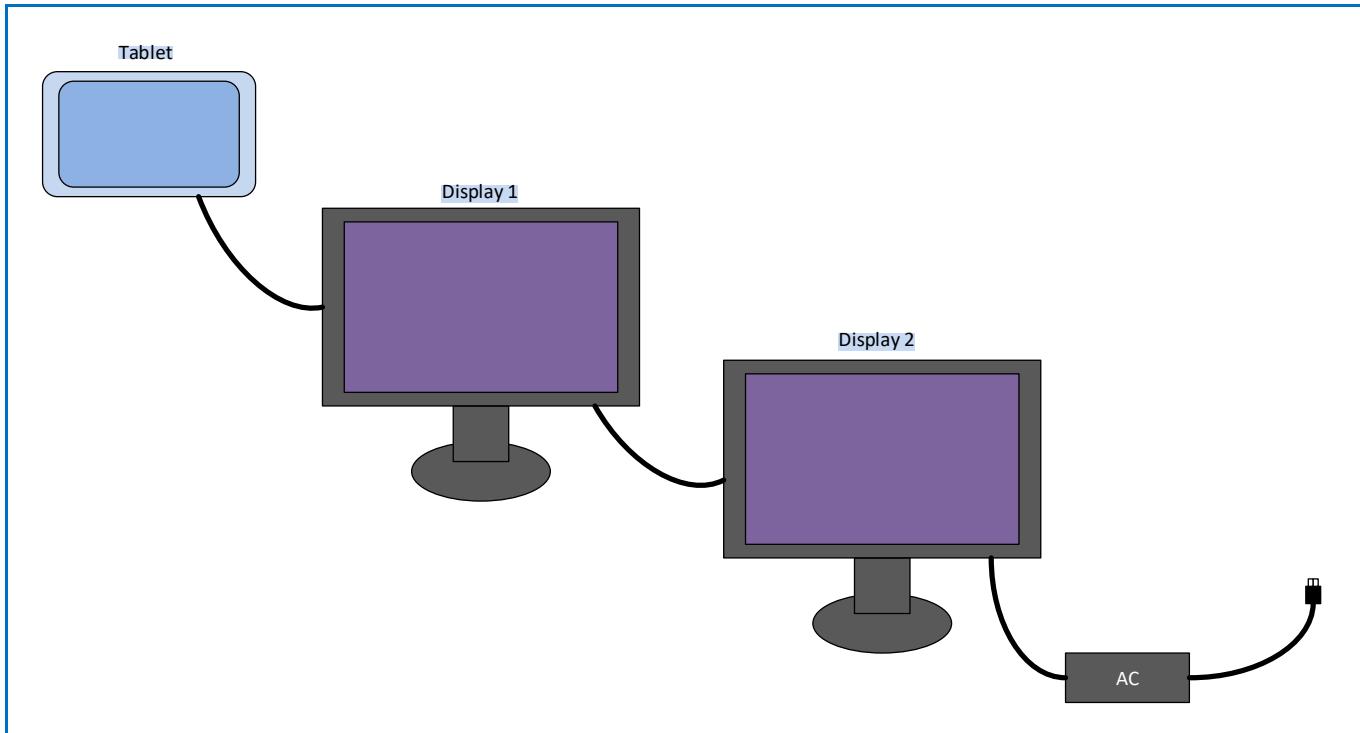
In the case of the external power source, the power can either be from an AC supply directly connected to the device or from an AC supply connected to an Attached device, which is also getting unconstrained power from its power supply. The Unconstrained Power bit is in this way communicated through a PD system indicating that the origin of the power is from a single or multiple AC supplies, from a battery bank, or similar:

- If the “Unconstrained Power” bit is set, then that power is originally sourced from an AC supply.
- Devices capable of consuming on multiple ports can only claim that they have “Unconstrained Power” for the power Advertised as a provider Port if there is unconstrained power beyond that needed for normal operation coming from external supplies, (e.g., multiple AC supplies).
- This concept applies as the power is routed through multiple provider and Consumer tiers, so, as an example. Power provided out of a monitor that is connected to a monitor that gets power from an AC supply, will claim it has “Unconstrained Power” even though it is not directly connected to the AC supply.

An example use case is a Tablet computer that is used with two USB A/V displays that are daisy chained (see [Figure 8-1 “Example of daisy chained displays”](#)). The tablet and 1st display are not externally powered, (meaning, they have no source of power outside of USB PD). The 2nd display has an external supply Attached which could either be a USB PD based supply or some other form of external supply. When the displays are connected as shown, the power adapter Attached to the 2nd display is able to power both the 1st display and the tablet. In this case the 2nd display will indicate the presence of a sufficiently sized wall wart to the 1st display, by setting its “Unconstrained Power” bit. The 1st display will then in turn assess and indicate the presence of the extra power to the tablet by setting its “Unconstrained Power” bit. Power is transmitted through the system to all devices, provided that there is sufficient power available from the external supply.



Figure 8-1 “Example of daisy chained displays”



Another example use case is a laptop computer that is attached to both an external supply and a Tablet computer. In this situation, if the external supply is large enough to power the laptop in its normal state as well as charge an external device, the laptop would set its “Unconstrained Power” bit and the tablet will allow itself to charge at its peak rate. If the external supply is small, however, and would not prevent the laptop from discharging if maximal power is drawn by the external device, the laptop would not set its “Unconstrained Power” bit, and the tablet can choose to draw less than what is offered. This amount could be just enough to prevent the tablet from discharging, or none at all. Alternatively, if the tablet determines that the laptop has significantly larger battery with more charge than the tablet has, the tablet can still choose to charge itself, although possibly not at the maximal rate.

In this way, Sinks that do not receive the "Unconstrained Power" bit from the connected Source can still choose to charge their batteries, or charge at a reduced rate, if their policy determines that the impact to the Source is minimal -- such as in the case of a phone with a small battery charging from a laptop with a large battery. These policies can be decided via further USB PD communication.

8.2.6.2 Battery Supplies

When monitored, and a USB interface is supported, the Battery state **Shall** be reported to the System Policy Manager using the USB interface.

If the device is battery-powered but is in a state that is primarily for charging external devices, the device is considered to be an unconstrained source of power and thus **Should** set the “Unconstrained Power” bit.

A simplified algorithm is detailed below to ensure that Battery powered devices will get charge from non-Battery powered devices when possible, and also to ensure that devices do not constantly Power Role Swap back and forth.

When two devices are connected that do not have Unconstrained Power, they **Should** define their own policies so as to prevent constant Power Role Swapping.

This algorithm uses the “Unconstrained Power” bit (see [Section 6.4.1.2.2.3 “Unconstrained Power”](#) and [Section 6.4.1.3.1.3 “Unconstrained Power”](#)), thus the decisions are based on the availability and sufficiency of an external supply, not the full capabilities of a system or device or product.

Recommendations:

- Provider/Consumers using large external sources (“Unconstrained Power” bit set) **Should** always deny Power Role Swap requests from Consumer/Providers not using external sources (“Unconstrained Power” bit cleared).
- Provider/Consumers not using large external sources (“Unconstrained Powered” bit cleared) **Should** always accept a Power Role Swap request from a Consumer/Provider using large external power sources (“Unconstrained Power” bit set) unless the requester is not able to provide the requirements of the present Provider/Consumer.

8.2.7 Interface to the Policy Engine

The Device Policy Manager **Shall** maintain an interface to the Policy Engine for each Port in the device.

8.2.7.1 Device Policy Manager in a Provider

The Device Policy Manager in a Provider **Shall** also provide the following functions to the Policy Engine:

- Inform the Policy Engine of changes in cable/ device Attachment status for a given cable.
- Inform the Policy Engine whenever the Source capabilities available for a Port change.
- Evaluate requests from an Attached Consumer and provide responses to the Policy Engine.
- Respond to requests for power supply transitions from the Policy Engine.
- Indication to Policy Engine when power supply transitions are complete.
- Maintain a Power Reserve for devices operating on a Port at less than maximum power.

8.2.7.2 Device Policy Manager in a Consumer

The Device Policy Manager in a Consumer **Shall** also provide the following functions to the Policy Engine:

- Inform the Policy Engine of changes in cable/device Attachment status.
- Inform the Policy Engine whenever the power requirements for a Port change.
- Evaluate Source capabilities and provide suitable responses:
 - Request from offered capabilities.
 - Indicate whether additional power is required.
- Respond to requests for Sink transitions from the Policy Engine.

8.2.7.3 Device Policy Manager in a Dual-Role Power Device

The Device Policy Manager in a Dual-Role Power Device **Shall** provide the following functions to the Policy Engine:

- Provider Device Policy Manager
- Consumer Device Policy Manager
- Interface for the Policy Engine to request power supply transitions from Source to Sink and vice versa.
- Indications to Policy Engine during Power Role Swap transitions.

8.2.7.4 Device Policy Manager in a Dual-Role Power Device Dead Battery handling

The Device Policy Manager in a Dual-Role Power Device with a Dead Battery **Should**:

- Switch Ports to Sink-only or Sinking DFP operation to obtain power from the next Attached Source.
- Use V_{BUS} from the Attached Source to power the USB Power Delivery communications as well as charging to enable the negotiation of higher input power.

8.3 Policy Engine

8.3.1 Introduction

There is one Policy Engine instance per Port that interacts with the Device Policy Manager in order to implement the present Local Policy for that particular Port. This section includes:

- Message sequences for various operations.
- State diagrams covering operation of Sources, Sinks and Cable Plugs.

8.3.2 Atomic Message Sequence Diagrams

8.3.2.1 Introduction

The Device Policy Engine drives the Message sequences and responses based on both the expected Message sequences and the present Local Policy.

An AMS **Shall** be defined as a Message sequence that starts and/or ends in either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** states (see [Section 8.3.3.2 "Policy Engine Source Port State Diagram"](#), [Section 8.3.3.3 "Policy Engine Sink Port State Diagram"](#) and [Section 8.3.3.26 "Cable Plug Specific State Diagrams"](#)).

In addition, the Cable Plug discovery sequence specified in Section 8.3.3.24.3 **Shall** be defined as an AMS.

The Source and Sink indicate to the Protocol Layer when an AMS starts and ends on entry to/exit from **PE_SRC_Ready** or **PE_SNK_Ready** (see [Section 8.3.3.2 "Policy Engine Source Port State Diagram"](#) and [Section 8.3.3.3 "Policy Engine Sink Port State Diagram"](#)).

An AMS **Shall** be considered to have been started by the initiator when the protocol engine signals the Policy engine that transmission is a success (the **GoodCRC** Message has been received in response to the initial message). For the receiving port the AMS **Shall** be considered to have started when the initial message has arrived.

An AMS **Shall** be considered to have ended:

- When the Protocol Engine signals the Policy Engine that transmission of the final Message in the AMS is a success and for the opposite port when the final Message has been received.
- A **Soft_Reset** Message, **Hard Reset** Signaling for SOP' or SOP'' or **Cable Reset** Signaling has been sent or received.

[Section 8.3.2.1.3 "Atomic Message Sequences"](#) gives details of these AMS's.

This section contains sequence diagrams that highlight some of the more interesting transactions. It is by no means a complete summary of all possible combinations but is illustrative in nature.

8.3.2.1.1 Basic Message Exchange

[Figure 8-2 "Basic Message Exchange \(Successful\)"](#) below illustrates how a Message is sent. [Table 8.1 "Basic Message Flow"](#) details the steps in the flow. Note that the sender might be either a Source or Sink while the receiver might be either a Sink or Source. The basic Message sequence is the same. It starts when the Message Sender's Protocol Layer at the behest of its Policy Engine forms a Message that it passes to the Physical Layer.

Figure 8-2 “Basic Message Exchange (Successful)”

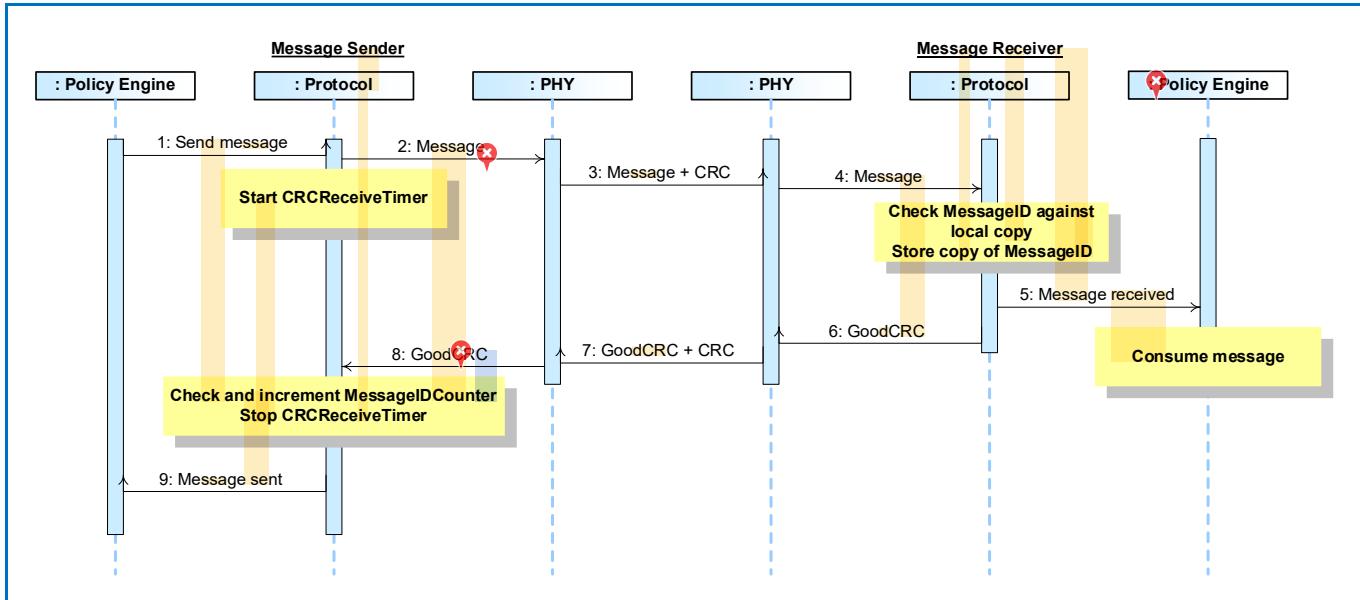


Table 8.1 “Basic Message Flow”

Step	Message Sender	Message Receiver
1	Policy Engine directs Protocol Layer to send a Message.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends a CRC and sends the Message. Starts CRCReceiveTimer .	Physical Layer receives the Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer forwards the received Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it to the Physical Layer.
7	Physical Layer receives the Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer. Protocol Layer checks and increments the MessageIDCounter and stops CRCReceiveTimer .	
9	Protocol Layer informs the Policy Engine that the Message was successfully sent.	

8.3.2.1.2 Errors in Basic Message flow

There are various points during the Message flow where failures in communication or other issues can occur. [Figure 8-3 “Basic Message flow indicating possible errors”](#) is an annotated version of [Figure 8-2 “Basic Message Exchange \(Successful\)”](#) indicating at which point issues can occur. [Table 8.2 “Potential issues in Basic Message Flow”](#) details the steps in the flow.

Figure 8-3 “Basic Message flow indicating possible errors”

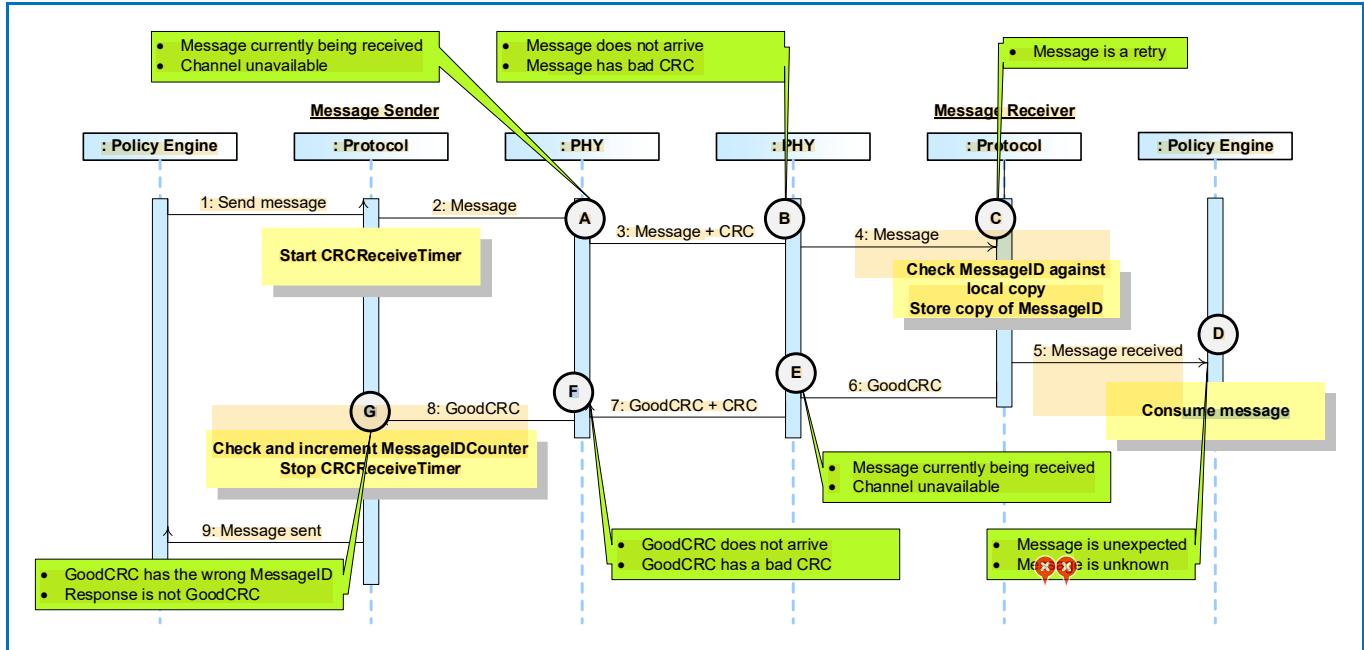


Table 8.2 “Potential issues in Basic Message Flow”

Point	Possible issues
A	<p>1) There is an incoming Message on the channel meaning that the PHY Layer is unable to send. In this case the outgoing Message is removed from the queue and the incoming Message processed.</p> <p>2) Due to some sort of noise on the line it is not possible to transmit. In this case the outgoing Message is Discarded by the PHY Layer. Retransmission is via the Protocol Layer’s normal mechanism.</p>
B	<p>1) Message does not arrive at the Physical Layer due to noise on the channel.</p> <p>2) Message arrives but has been corrupted and has a bad CRC.</p> <p>There is no Message to pass up to the Protocol Layer on the receiver which means a GoodCRC Message is not sent. This leads to a CRCReceiveTimer timeout in the Message Sender.</p>
C	<p>MessageID of received Message matches stored MessageID so this is a retry. Message is not passed up to the Policy Engine.</p>
D	<p>1) Policy Engine receives a known Message that it was not expecting.</p> <p>2) Policy Engine receives an Unrecognized Message.</p> <p>These cases are errors in the protocol which could lead to the generation of a Soft_Reset Message.</p>
E	Same as point A but at the Message Receiver side.
F	<p>1) GoodCRC Message response does not arrive at the Message Sender side due to the noise on the channel.</p> <p>2) GoodCRC Message response arrives but has a bad CRC.</p> <p>A GoodCRC Message is not received by the Message Sender’s Protocol Layer. This leads to a CRCReceiveTimer timeout in the Message Sender.</p>
G	<p>1) GoodCRC Message is received but does contain the same MessageID as the transmitted Message.</p> <p>2) A Message is received but it is not a GoodCRC Message (similar case to that of an unexpected or unknown Message but this time detected in the Protocol Layer).</p> <p>Both of these issues indicate errors in receiving an expected GoodCRC Message which will lead to a CRCReceiveTimer timeout in the Protocol Layer and a subsequent retry (except for communications with Cable Plugs).</p>

Figure 8-4 “Basic Message Flow with Bad CRC followed by a Retry” illustrates one of these cases; the basic Message flow with a retry due to a bad CRC at the Message Receiver. It starts when the Message Sender’s Protocol Layer at the behest of its Policy Engine forms a Message that it passes to the Physical Layer. The Protocol Layer is responsible for retries on a “n’ strikes and you are out” basis (*nRetryCount*). **Table 8.3 “Basic Message Flow with CRC failure”** details the steps in the flow.

Figure 8-4 “Basic Message Flow with Bad CRC followed by a Retry”

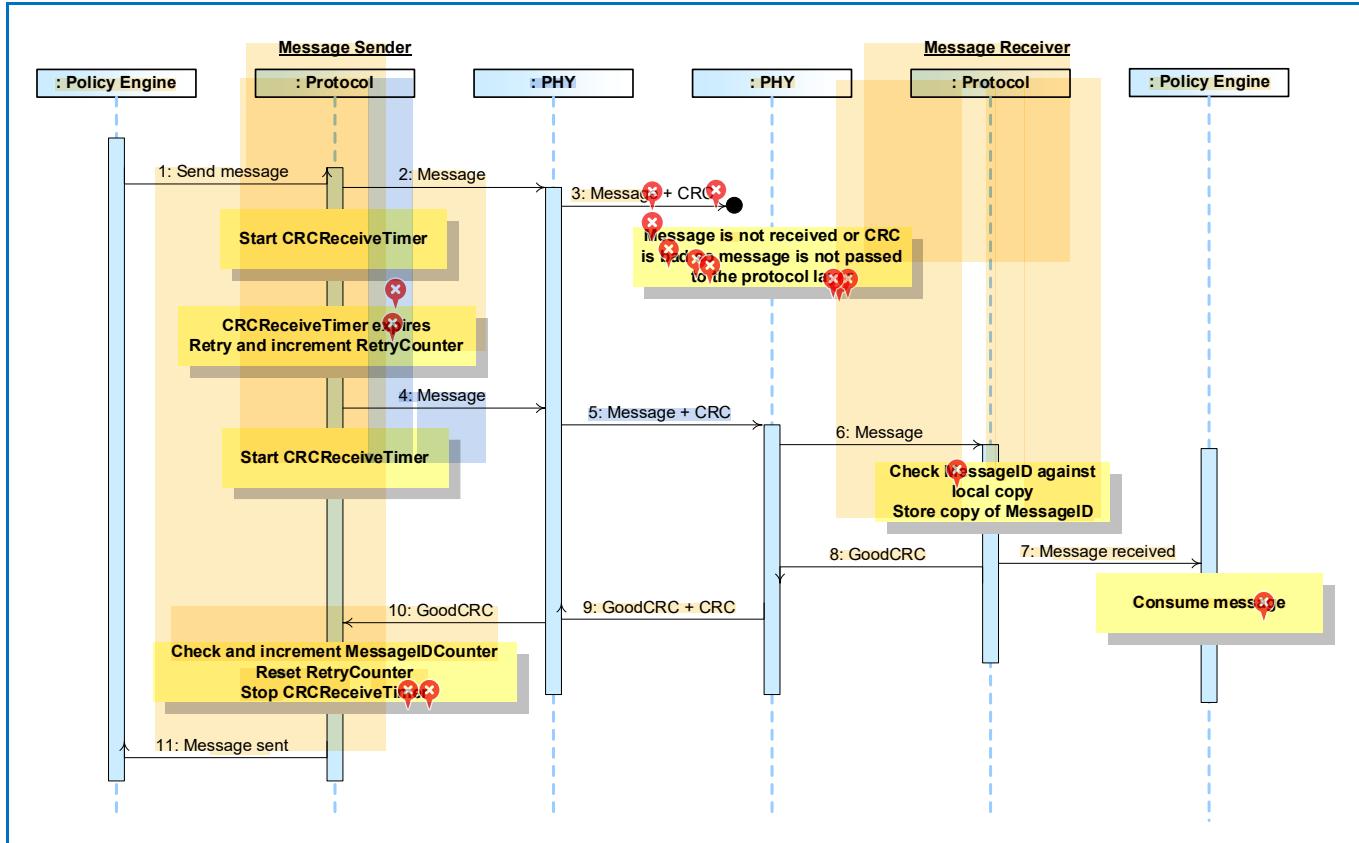


Table 8.3 “Basic Message Flow with CRC failure”

Step	Message Sender	Message Receiver
1	Policy Engine directs Protocol Layer to send a Message.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends a CRC and sends the Message. Starts <i>CRCReceiveTimer</i> . 	Physical Layer receives no Message or a Message with an incorrect CRC. Nothing is passed to Protocol Layer.
4	Since no response is received, the <i>CRCReceiveTimer</i> will expire and trigger the first retry by the Protocol Layer. The <i>RetryCounter</i> is incremented. Protocol Layer passes the Message to the Physical Layer.	
5	Physical Layer appends a CRC and sends the Message. Starts <i>CRCReceiveTimer</i> . 	Physical Layer receives the Message and checks the CRC to verify the Message.
6		Physical Layer removes the CRC and forwards the Message to the Protocol Layer.
7		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer forwards the received Message information to the Policy Engine that consumes it.
8		Protocol Layer generates a <i>GoodCRC</i> Message and passes it to the Physical Layer.
9	Physical Layer receives the Message and checks the <i>CRC</i> to verify the Message. 	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
10	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.  	
11	Protocol Layer verifies the <i>MessageID</i> , stops <i>CRCReceiveTimer</i> and resets the <i>RetryCounter</i> . Protocol Layer informs the Policy Engine that the Message was successfully sent.	

8.3.2.1.3 Atomic Message Sequences

The types of Atomic Message Sequences (AMS) are listed in [Table 8.4 “Atomic Message Sequences”](#). The following tables list sequences of either Messages or combinations of Messages and one or more embedded AMSes which are Non-Interruptible. Where there is an embedded AMS the entire Message Sequence is treated as an AMS and the R_p value used for collision avoidance (see [Section 5.7 “Collision Avoidance”](#)) **Shall** only be changed on leaving or entering the ready state at the beginning or end of the entire Message Sequence, and not at the start or end of the embedded AMS. Note that an AMS is has not started until the first Message in the sequence has been successfully sent (i.e., a **GoodCRC** Message has been received acknowledging the Message).

[Table 8.32 “AMS: Hard Reset”](#) details a Hard Reset (which is Signaling not an AMS) followed by an SPR Contract Negotiation AMS which **Shall** be treated as Non-Interruptible.

Table 8.4 “Atomic Message Sequences”

Type of AMS	Table Reference	Section Reference
Power Negotiation (SPR)	Table 8.5 “AMS: Power Negotiation (SPR)”	Section 8.3.2.2.1
Power Negotiation (EPR)	Table 8.6 “AMS: Power Negotiation (EPR)”	Section 8.3.2.2.2
Unsupported Message	Table 8.7 “AMS: Unsupported Message”	Section 8.3.2.3
Ping	Table 8.8 “AMS: Ping”	Section 8.3.2.40
Soft Reset	Table 8.9 “AMS: Soft Reset”	Section 8.3.2.5
Data Reset	Table 8.10 “AMS: Data Reset”	Section 8.3.2.6
Hard Reset	Table 8.32 “AMS: Hard Reset”	Section 8.3.2.7
Power Role Swap	Table 8.11 “AMS: Power Role Swap”	Section 8.3.2.8
Fast Role Swap	Table 8.12 “AMS: Fast Role Swap”	Section 8.3.2.9
Data Role Swap	Table 8.12 “AMS: Fast Role Swap”	Section 8.3.2.10
VCONN Swap	Table 8.14 “AMS: VCONN Swap”	Section 8.3.2.11
Alert	Table 8.15 “AMS: Alert”	Section 8.3.2.12.1
Status	Table 8.16 “AMS: Status”	Section 8.3.2.12.2
Source/Sink Capabilities (SPR)	Table 8.17 “AMS: Source/Sink Capabilities (SPR)”	Section 8.3.2.12.3.1
Source/Sink Capabilities (EPR)	Table 8.18 “AMS: Source/Sink Capabilities (EPR)”	Section 8.3.2.12.3.2
Extended Capabilities	Table 8.19 “AMS: Extended Capabilities”	Section 8.3.2.12.4
Battery Capabilities and Status	Table 8.20 “AMS: Battery Capabilities”	Section 8.3.2.12.5
Manufacturer Information	Table 8.21 “AMS: Manufacturer Information”	Section 8.3.2.12.6
Country Codes	Table 8.22 “AMS: Country Codes”	Section 8.3.2.12.7
Country Information	Table 8.23 “AMS: Country Information”	Section 8.3.2.12.8
Revision Information	Table 8.24 “AMS: Revision Information”	Section 8.3.2.12.9
Source Information	Table 8.25 “AMS: Source Information”	Section 8.3.2.12.10
Security	Table 8.26 “AMS: Security”	Section 8.3.2.13
Firmware Update	Table 8.27 “AMS: Firmware Update”	Section 8.3.2.14
Structured VDM	Table 8.28 “AMS: Structured VDM”	Section 8.3.2.15
Built-In Self-Test (BIST)	Table 8.29 “AMS: Built-In Self-Test (BIST)”	Section 8.3.2.16
Enter USB	Table 8.30 “AMS: Enter USB”	Section 8.3.2.17
Unstructured VDM	Table 8.31 “AMS: Unstructured VDM”	Section 8.3.2.18

8.3.2.1.3.1

AMS: Power Negotiation (SPR)

Table 8.5 “AMS: Power Negotiation (SPR)”

AMS	Interruptible	Message Sequence	Conditions	AMS Ref
SPR Explicit Contract Negotiation (Accept)	1) <i>Source_Capabilities</i> Message 2) <i>Request</i> Message 3) <i>Accept</i> Message 4) <i>PS_RDY</i> Message	Started by  Source, SPR Mode	Section 8.3.2.2.1.1.1	Section 8.3.3.2 , Section 8.3.3.3
SPR Explicit Contract Negotiation (Reject)	1) <i>Source_Capabilities</i> Message 2) <i>Request</i> Message 3) <i>Reject</i> Message		Section 8.3.2.2.1.1.2	
SPR Explicit Contract Negotiation (Wait)	1) <i>Source_Capabilities</i> Message 2) <i>Request</i> Message 3) <i>Wait</i> Message		Section 8.3.2.2.1.1.3	
Reclaiming Power with GotoMin Message	 4) <i>GotoMin</i> Message 5) <i>PS_RDY</i> Message		Section 8.3.2.2.1.2	
SPR PPS Keep Alive	1) <i>Request</i> Message 2) <i>Accept</i> Message 3) <i>PS_RDY</i> Message	Started by Sink, SPR Mode	Section 8.3.2.2.1.3	Section 8.3.3.3
SPR Sink Makes Request (Accept)	1) <i>Request</i> Message 2) <i>Accept</i> Message 3) <i>PS_RDY</i> Message		Section 8.3.2.2.1.4.1	Section 8.3.3.2 , Section 8.3.3.3
SPR Sink Makes Request (Reject)	1) <i>Request</i> Message 2) <i>Reject</i> Message		Section 8.3.2.2.1.4.2	
SPR Sink Makes Request (Wait)	1) <i>Request</i> Message 2) <i>Wait</i> Message		Section 8.3.2.2.1.4.30	

8.3.2.1.3.2

AMS: Power Negotiation (EPR)

Table 8.6 “AMS: Power Negotiation (EPR)”

AMS	Interruptible	Message Sequence	Conditions	AMS Ref
Entering EPR Mode (Success)	1) EPR_Mode (Enter) Message 2) EPR_Mode (Enter Acknowledge) Message 3) Vconn Source Swap, initiated by non- Vconn Source (Accept) AMS (<i>Optional</i>). 4) Initiator to Responder Discover Identity (ACK) AMS (<i>Optional</i> for Sources with captive cables) 5) EPR_Mode (Enter Succeeded) Message 6) EPR Explicit Contract Negotiation AMS	Started by Sink, SPR Mode	Section 8.3.2.2.2.1 Section 8.3.2.9.1 Section 8.3.2.9.2 Section 8.3.2.13.3 Section 8.3.2.2.2.4	Section 8.3.3.26.1 , Section 8.3.3.26.2 , Section 8.3.3.20 , Section 8.3.3.21.1 , Section 8.3.3.22.1 , Section 8.3.3.2 , Section 8.3.3.3
Entering EPR Mode (Failure due to non-EPR cable)	1) EPR_Mode (Enter) Message 2) EPR_Mode (Enter Acknowledge) Message 3) Vconn Source Swap, initiated by non- Vconn Source (Accept) AMS(<i>Optional</i>). 4) Initiator to Responder Discover Identity (ACK) AMS (<i>Optional</i> for Sources with captive cables) 5) EPR_Mode (Enter Failed) Message	Started by Sink, SPR Mode	Section 8.3.2.2.2.2 Section 8.3.2.9.1 Section 8.3.2.9.2 Section 8.3.2.13.3	Section 8.3.3.26.1 , Section 8.3.3.26.2 , Section 8.3.3.20 , Section 8.3.3.21.1 , Section 8.3.3.22.1
Entering EPR Mode (Failure of VCONN Swap)	1) EPR_Mode (Enter) Message. 2) EPR_Mode (Enter Acknowledge) Message. 3) Vconn Source Swap, initiated by non- Vconn Source (Reject) AMS(<i>Optional</i>). 4) EPR_Mode (Enter Failed) Message	Started by Sink, SPR Mode	Section 8.3.2.2.2.3 Section 8.3.2.9.1 Section 8.3.2.9.2	Section 8.3.3.26.1 , Section 8.3.3.26.2 , Section 8.3.3.20
📍 EPR Explicit Contract Negotiation (Accept)	1) EPR_Source_Capabilities Message 2) EPR_Request Message 3) Accept Message 4) PS_RDY Message📍	Started by Source, EPR Mode	Section 8.3.2.2.2.1	Section 8.3.3.2 , Section 8.3.3.3

EPR Explicit Contract Negotiation (Reject)	1) EPR_Source_Capabilities Message 2) EPR_Request Message 3) Reject Message		Section 8.3.2.2.2.2.2	
EPR Explicit Contract Negotiation (Wait)	1) EPR_Source_Capabilities Message 2) EPR_Request Message 3) Wait Message		Section 8.3.2.2.2.2.30	
EPR Keep Alive	1) EPR_KeepAlive Message 2) EPR_KeepAlive_Ack Message	Started by Sink, EPR Mode	Section 8.3.2.2.2.5	
Exiting EPR Mode (Sink Initiated)	1) EPR_Mode (Exit) Message 2) SPR Explicit Contract Negotiation AMS	Started by Sink, EPR Mode	Section 8.3.2.2.2.6 Section 8.3.2.2.1.1	Section 8.3.3.26.3 , Section 8.3.3.26.4 , Section 8.3.3.2 , Section 8.3.3.3
✖ Exiting EPR Mode (Source Initiated)	1) EPR_Mode (Exit) Message 2) SPR Explicit Contract Negotiation AMS	Started by Source, EPR Mode	Section 8.3.2.2.2.7 Section 8.3.2.2.1.1	
EPR Sink Makes Request (Accept)	1) EPR_Request Message 2) Accept Message 3) PS_RDY Message	Started by Sink, EPR Mode	Section 8.3.2.2.2.5	Section 8.3.3.2 , Section 8.3.3.3
EPR Sink Makes Request (Reject)	1) EPR_Request Message 2) Reject Message	Started by Sink, EPR Mode	Section 8.3.2.2.2.5.2	
EPR Sink Makes Request (Wait)	1) EPR_Request Message 2) Wait Message	Started by Sink, EPR Mode	Section 8.3.2.2.2.5.3	

8.3.2.1.3.3

AMS: Unsupported Message

Table 8.7 “AMS: Unsupported Message”

AMS	Interruptible	Message Sequence	Conditions	AMS Ref
✖ Unsupported Message	1) Any Message which is not supported by the Source or Sink 2) Not_Supported Message	Started by Source or Sink	Section 8.3.2.3	Section 8.3.3.6.2

8.3.2.1.3.4

AMS: Ping

Table 8.8 “AMS: Ping”

AMS	Interruptible	Message Sequence	Conditions	AMS Ref
Ping	1) <i>Ping</i> Message	Started by Source	<i>Section 8.3.2.4</i>	<i>Section 8.3.3.7</i>

8.3.2.1.3.5

AMS: Soft Reset

Table 8.9 “AMS: Soft Reset”

AMS	Interruptible	Message Sequence	Conditions	AMS Ref
Soft Reset	<ul style="list-style-type: none"> 1) <i>Soft_Reset</i> Message 2) <i>Accept</i> Message 3) In SPR Mode: SPR Explicit Contract Negotiation AMS or in EPR Mode: 4) EPR Explicit Contract Negotiation AMS. 	Started by Source or Sink	<i>Section 8.3.2.5</i>    <i>Section 8.3.2.1.1</i> <i>Section 8.3.2.2.4</i>	<i>Section 8.3.3.4.1</i> , <i>Section 8.3.3.4.2</i> , <i>Section 8.3.3.25.2.1</i> , <i>Section 8.3.3.25.2.3</i> , <i>Section 8.3.3.25.2.4</i> <i>Section 8.3.3.2</i> , <i>Section 8.3.3.3</i>

8.3.2.1.3.6

AMS: Data Reset

Table 8.10 “AMS: Data Reset”

AMS	Interruptible	Message Sequence	Conditions	AMS Ref
DFP Initiated Data Reset where the DFP is the VCONN Source	1) <i>Data_Reset</i> Message 2) <i>Accept</i> Message 3) <i>Data_Reset_Complete</i> Message	Started by DFP	<i>Section 8.3.2.6.1</i>	<i>Section 8.3.3.5.1</i> , <i>Section 8.3.3.5.2</i>
DFP Receives Data Reset where the DFP is the VCONN Source	1) <i>Data_Reset</i> Message 2) <i>Accept</i> Message 3) <i>Data_Reset_Complete</i> Message	Started by UFP	<i>Section 8.3.2.6.2</i>	
DFP Initiated Data Reset where the UFP is the VCONN Source	1) <i>Data_Reset</i> Message 2) <i>Accept</i> Message 3) <i>PS_RDY</i> Message 4) <i>Data_Reset_Complete</i> Message	Started by DFP	<i>Section 8.3.2.6.3</i>	
DFP Receives Data Reset where the UFP is the VCONN Source	1) <i>Data_Reset</i> Message 2) <i>Accept</i> Message 3) <i>PS_RDY</i> Message 4) <i>Data_Reset_Complete</i> Message	Started by UFP	<i>Section 8.3.2.6.4</i>	

8.3.2.1.3.7

AMS: Power Role Swap

Table 8.11 “AMS: Power Role Swap”

AMS	Interruptible	Message Sequence	Conditions	AMS Ref
Source Initiated Power Role Swap (Accept)	1) <i>PR_Swap</i> Message 2) <i>Accept</i> Message 3) <i>PS_RDY</i> Message 4) PS_RDY Message 5) SPR Explicit Contract Negotiation AMS	Started by Source	<i>Section 8.3.2.8.1.1</i>	<i>Section 8.3.3.19.3, Section 8.3.3.19.4</i>
Source Initiated Power Role Swap (Reject)	1) <i>PR_Swap</i> Message 2) <i>Reject</i> Message		Section 8.3.2.8.1.2	
Source Initiated Power Role Swap (Wait)	1) <i>PR_Swap</i> Message 2) <i>Wait</i> Message		<i>Section 8.3.2.8.1.3</i>	
Sink Initiated Power Role Swap (Accept)	1) <i>PR_Swap</i> Message 2) <i>Accept</i> Message 3) <i>PS_RDY</i> Message 4) PS_RDY Message 5) SPR Explicit Contract Negotiation AMS	Started by Sink	<i>Section 8.3.2.8.2.1</i>	
Sink Initiated Power Role Swap (Reject)	1) <i>PR_Swap</i> Message 2) <i>Reject</i> Message		<i>Section 8.3.2.8.2.20</i>	
Sink Initiated Power Role Swap (Wait)	1) <i>PR_Swap</i> Message 2) <i>Wait</i> Message		<i>Section 8.3.2.8.2.30</i>	

8.3.2.1.3.8

AMS: Fast Role Swap

Table 8.12 “AMS: Fast Role Swap”

AMS	Interruptible	Message Sequence	Conditions	AMS Ref
Fast Role Swap	1) <i>FR_Swap</i> Message 2) <i>Accept</i> Message 3) <i>PS_RDY</i> Message 4) <i>PS_RDY</i> Message 5) SPR Explicit Contract Negotiation AMS	Started by Sink	<i>Section 8.3.2.9, Section 8.3.2.2.1.1</i>	<i>Section 8.3.3.2, Section 8.3.3.3, Section 8.3.3.20.5, Section 8.3.3.20.6</i>

8.3.2.1.3.9

AMS: Data Role Swap

Table 8.13 “AMS: Data Role Swap”

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
Data Role Swap, Initiated by UFP Operating as Sink (Accept)	1) <i>DR_Swap</i> Message 2) <i>Accept</i> Message	Started by Sink	<i>Section 8.3.2.10.1.1</i>	<i>Section 8.3.3.20.1</i> <i>Section 8.3.3.19.2</i>
Data Role Swap, Initiated by UFP Operating as Sink (Reject)	1) <i>DR_Swap</i> Message 2) <i>Reject</i> Message		<i>Section 8.3.2.10.1.2</i>	
Data Role Swap, Initiated by UFP Operating as Sink (Wait)	1) <i>DR_Swap</i> Message 2) <i>Wait</i> Message		<i>Section 8.3.2.10.1.3</i>	
Data Role Swap, Initiated by UFP Operating as Source (Accept)	1) <i>DR_Swap</i> Message 2) <i>Accept</i> Message	Started by Source	<i>Section 8.3.2.10.2.1</i>	
Data Role Swap, Initiated by UFP Operating as Source (Reject)	1) <i>DR_Swap</i> Message 2) <i>Reject</i> Message		<i>Section 8.3.2.10.2.2</i>	
Data Role Swap, Initiated by UFP Operating as Source (Wait)	1) <i>DR_Swap</i> Message 2) <i>Wait</i> Message		<i>Section 8.3.2.10.2.3</i>	
Data Role Swap, Initiated by DFP Operating as Source (Accept)	1) <i>DR_Swap</i> Message 2) <i>Accept</i> Message	Started by Source	<i>Section 8.3.2.10.3.1</i>	
Data Role Swap, Initiated by DFP Operating as Source (Reject)	1) <i>DR_Swap</i> Message 2) <i>Reject</i> Message		<i>Section 8.3.2.10.3.2</i>	
Data Role Swap, Initiated by DFP Operating as Source (Wait)	1) <i>DR_Swap</i> Message 2) <i>Wait</i> Message		<i>Section 8.3.2.10.3.3</i>	
Data Role Swap, Initiated by DFP Operating as Sink (Accept)	1) <i>DR_Swap</i> Message 2) <i>Accept</i> Message	Started by Sink	<i>Section 8.3.2.10.4.1</i>	
Data Role Swap, Initiated by DFP Operating as Sink (Reject)	1) <i>DR_Swap</i> Message 2) <i>Reject</i> Message		<i>Section 8.3.2.10.4.2</i>	
Data Role Swap, Initiated by DFP Operating as Sink (Wait)	1) <i>DR_Swap</i> Message 2) <i>Wait</i> Message		<i>Section 8.3.2.10.4.3</i>	

8.3.2.1.3.10

AMS: VCONN Swap

Table 8.14 "AMS: VCONN Swap"

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
Vconn Source Swap, initiated by Vconn Source (Accept)	1) VCONN_Swap Message 2) Accept Message 3) PS_RDY Message	Started by VCONN Source	Section 8.3.2.11.1.1	Section 8.3.3.21
Vconn Source Swap, initiated by Vconn Source (Reject)	1) VCONN_Swap Message 2) Reject Message		Section 8.3.2.11.1.2	
Vconn Source Swap, initiated by Vconn Source (Wait)	1) VCONN_Swap Message 2) Wait Message		Section 8.3.2.11.1.3	
Vconn Source Swap, initiated by non-Vconn Source (Accept)	1) VCONN_Swap Message 2) Accept Message 3) PS_RDY Message	Started by non-VCONN Source	Section 8.3.2.11.2.1	
Vconn Source Swap, initiated by non-Vconn Source (Reject)	1) VCONN_Swap Message 2) Reject Message		Section 8.3.2.11.2.20	
Vconn Source Swap, initiated by non-Vconn Source (Wait)	1) VCONN_Swap Message 2) Wait Message		Section 8.3.2.11.2.3	

8.3.2.1.3.11

AMS: Alert

Table 8.15 "AMS: Alert"

AMS	Message Sequence	Conditions	AMS Ref	AMS Ref
Source sends Alert to a Sink (SenderResponseTimer Timeout)	1) Alert Message	Started by Source	Section 8.3.2.10.1.1	Section 8.3.3.8.1 Section 8.3.3.8.2
Source sends Alert to a Sink (Get_Status Message)	1) Alert Message 2) Sink Gets Source Status AMS			
Sink sends Alert to a Source (SenderResponseTimer Timeout)	1) Alert Message	Started by Sink	Section 8.3.2.10.1.2	Section 8.3.3.8.3 Section 8.3.3.8.4
Sink sends Alert to a Source (Get_Status Message)	1) Alert Message 2) Source Gets Sink Status AMS			

8.3.2.1.3.12

AMS: Status

Table 8.16 “AMS: Status”

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
Sink Gets Source Status	1) <i>Get_Status</i> Message 2) <i>Status</i> Message	Started by Sink Started by Source	<i>Section</i> 8.3.2.10.2.1 <i>Section</i> 8.3.2.10.2.2	<i>Section</i> 8.3.3.10.1 , <i>Section</i> 8.3.3.10.2
Source Gets Sink Status	1) <i>Get_Status</i> Message 2) <i>Status</i> Message			
VCONN Source Gets Cable Plug Status	1) <i>Get_Status</i> Message 2) <i>Status</i> Message	Started by VCONN Source Started by Sink	<i>Section</i> 8.3.2.10.2.3 <i>Section</i> 8.3.2.10.2.4	<i>Section</i> 8.3.3.10.3 , <i>Section</i> 8.3.3.10.4
Sink Gets Source PPS Status	1) <i>Get_PPS_Status</i> Message 2) <i>PPS_Status</i> Message			

8.3.2.1.3.13

AMS: Source/Sink Capabilities (SPR)

Table 8.17 “AMS: Source/Sink Capabilities (SPR)”

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
Sink Gets Source Capabilities (EPR Mode)	1) <i>Get_Source_Cap</i> Message 2) <i>Source_Capabilities</i> Message 	Started by Sink	<i>Section 8.3.2.12.3.1.1</i> <i>Section 8.3.2.2.1.4.1</i> <i>Section 8.3.2.2.1.4.2</i> <i>Section 8.3.2.2.1.4.3</i>	<i>Section 8.3.3.2</i> , <i>Section 8.3.3.3</i>
Sink Gets Source Capabilities (Accept in SPR Mode)	1) <i>Get_Source_Cap</i> Message 2) <i>Source_Capabilities</i> Message 3) In SPR Mode only:  4) SPR Sink Makes Request   5) SPR Sink Makes Request (Accept) AMS			
Sink Gets Source Capabilities (Reject in SPR Mode)	1) <i>Get_Source_Cap</i> Message 2) <i>Source_Capabilities</i> Message 3) In SPR Mode only: SPR Sink Makes Request (Reject) AMS			
Sink Gets Source Capabilities (Wait in SPR Mode)	1) <i>Get_Source_Cap</i> Message 2) <i>Source_Capabilities</i> Message 3) In SPR Mode only: 4) SPR Sink Makes Request (Wait) AMS			
Dual-Role Source Gets Source Capabilities from a Dual-Role Sink	1) <i>Get_Source_Cap</i> Message 2) <i>Source_Capabilities</i> Message	Started by Source	<i>Section 8.3.2.12.3.1.2</i>	<i>Section 8.3.3.19.7</i> , <i>Section 8.3.3.19.10</i>
Source Gets Sink Capabilities	1) <i>Get_Sink_Cap</i> Message 2) <i>Sink_Capabilities</i> Message	Started by Source	<i>Section 8.3.2.12.3.1.3</i>	<i>Section 8.3.3.2</i> , <i>Section 8.3.3.3</i>
Dual-Role Sink Get Sink Capabilities from a Dual-Role Source	1) <i>Get_Sink_Cap</i> Message 2) <i>Sink_Capabilities</i> Message	Started by Sink	<i>Section 8.3.2.12.3.1.4</i>	<i>Section 8.3.3.19.9</i> , <i>Section 8.3.3.19.8</i>

8.3.2.1.3.14

AMS: Source/Sink Capabilities (EPR)

Table 8.18 “AMS: Source/Sink Capabilities (EPR)”

AMS	Interruptible	Message Sequence	Conditions	AMS Ref
Sink Gets EPR Source Capabilities (SPR Mode)	1) <i>EPR_Get_Source_Cap</i> Message 2) <i>EPR_Source_Capabilities</i> Message	Started by Sink	<i>Section 8.3.2.12.3.2.1</i> <i>Section 8.3.2.2.2.5.1</i> <i>Section 8.3.2.2.2.5.20</i> <i>Section 8.3.2.2.2.5.30</i>	<i>Section 8.3.3.2,</i> <i>Section 8.3.3.3</i>
Sink Gets EPR Source Capabilities (Accept in EPR Mode)	1) <i>EPR_Get_Source_Cap</i> Message 2) <i>EPR_Source_Capabilities</i> Message 3) In EPR Mode only: EPR Sink Makes Request 4) EPR Sink Makes Request (Accept) AMS			
Sink Gets EPR Source Capabilities (Reject in EPR Mode)	1) <i>EPR_Get_Source_Cap</i> Message 2) <i>EPR_Source_Capabilities</i> Message 3) In EPR Mode only: 4) EPR Sink Makes Request (Reject) AMS			
Sink Gets EPR Source Capabilities (Wait in EPR Mode)	1) <i>EPR_Get_Source_Cap</i> Message 2) <i>EPR_Source_Capabilities</i> Message 3) In EPR Mode only: 4) EPR Sink Makes Request (Wait) AMS			
Dual-Role Source Gets Source Capabilities from a Dual-Role EPR Sink	1) <i>EPR_Get_Source_Cap</i> Message 2) <i>EPR_Source_Capabilities</i> Message	Started by Source	<i>Section 8.3.2.12.3.2.2</i>	<i>Section 8.3.3.19.7,</i> <i>Section 8.3.3.19.10</i>
Source Gets Sink EPR Capabilities	1) <i>EPR_Get_Sink_Cap</i> Message 2) <i>EPR_Sink_Capabilities</i> Message	Started by Source	<i>Section 8.3.2.12.3.2.3</i>	<i>Section 8.3.3.2,</i> <i>Section 8.3.3.3</i>
Dual-Role Sink Get Sink EPR Capabilities from a Dual-Role Source	1) <i>EPR_Get_Sink_Cap</i> Message 2) <i>EPR_Sink_Capabilities</i> Message	Started by Sink	<i>Section 8.3.2.12.3.2.4</i>	<i>Section 8.3.3.19.9,</i> <i>Section 8.3.3.19.8</i>

8.3.2.1.3.15

AMS: Extended Capabilities

Table 8.19 “AMS: Extended Capabilities”

AMS	Interruptible	Message Sequence	Conditions	AMS Ref
Sink Gets Source Extended Capabilities	1) <i>Get_Source_Cap_Extended</i> Message 2) <i>Source_Capabilities_Extended</i> Message	Started by Sink	<i>Section 8.3.2.12.4.1</i>	<i>Section 8.3.3.9.1</i> , <i>Section 8.3.3.9.2</i>
Dual-Role Source Gets Source Capabilities Extended from a Dual-Role Sink	1) <i>Get_Source_Cap_Extended</i> Message 2) <i>Source_Capabilities_Extended</i> Message	Started by Source	<i>Section 8.3.2.12.4.2</i>	<i>Section 8.3.3.19.11</i> , <i>Section 8.3.3.19.12</i>
Source Gets Sink Extended Capabilities	1) <i>Get_Sink_Cap_Extended</i> Message 2) <i>Sink_Capabilities_Extended</i> Message	Started by Source	<i>Section 8.3.2.12.4.3</i>	<i>Section 8.3.3.9.3</i> , <i>Section 8.3.3.9.4</i>
Dual-Role Sink Gets Sink Capabilities Extended from a Dual-Role Source	1) <i>Get_Sink_Cap_Extended</i> Message 2) <i>Sink_Capabilities_Extended</i> Message	Started by Sink	<i>Section 8.3.2.12.4.4</i>	<i>Section 8.3.3.19.13</i> , <i>Section 8.3.3.19.14</i>

8.3.2.1.3.16

AMS: Battery Capabilities

Table 8.20 “AMS: Battery Capabilities”

AMS	Interruptible	Message Sequence	Conditions	AMS Ref
Sink Gets Battery Capabilities	1) <i>Get_Battery_Cap</i> Message 2) <i>Battery_Capabilities</i> Message	Started by Sink	<i>Section 8.3.2.12.5.1</i>	<i>Section 8.3.3.11.1</i> , <i>Section 8.3.3.11.2</i>
Source Gets Battery Capabilities	1) <i>Get_Battery_Cap</i> Message 2) <i>Battery_Capabilities</i> Message	Started by Source	<i>Section 8.3.2.12.5.2</i>	
Sink Gets Battery Status	1) <i>Get_Battery_Status</i> Message 2) <i>Battery_Status</i> Message	Started by Sink	<i>Section 8.3.2.12.5.3</i>	<i>Section 8.3.3.12.1</i> , <i>Section 8.3.3.12.2</i>
Sink Gets Battery Capabilities	1) <i>Get_Battery_Cap</i> Message 2) <i>Battery_Capabilities</i> Message	Started by Sink	<i>Section 8.3.2.12.5.4</i>	

8.3.2.1.3.17

AMS: Manufacturer Information

Table 8.21 “AMS: Manufacturer Information”

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
Source Gets Port Manufacturer Information from a Sink	1) <i>Get_Manufacturer_Info</i> Message 2) <i>Manufacturer_Info</i> Message	Started by Source	Section 8.3.2.12.6.1	Section 8.3.3.13.1 , Section 8.3.3.13.2
Sink Gets Port Manufacturer Information from a Source	1) <i>Get_Manufacturer_Info</i> Message 2) <i>Manufacturer_Info</i> Message	Started by Sink	Section 8.3.2.12.6.2	
Source Gets Battery Manufacturer Information from a Sink	1) <i>Get_Manufacturer_Info</i> Message 2) <i>Manufacturer_Info</i> Message	Started by Source	Section 8.3.2.12.6.3	
Sink Gets Battery Manufacturer Information from a Source	1) <i>Get_Manufacturer_Info</i> Message 2) <i>Manufacturer_Info</i> Message	Started by Sink	Section 8.3.2.12.6.4	
VCONN Source Gets Manufacturer Information from a Cable Plug	1) <i>Get_Manufacturer_Info</i> Message 2) <i>Manufacturer_Info</i> Message	Started by VCONN Source	Section 8.3.2.12.6.5	

8.3.2.1.3.18

AMS: Country Codes

Table 8.22 “AMS: Country Codes”

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
Source Gets Country Codes from a Sink	1) <i>Get_Country_Codes</i> Message 2) <i>Country_Codes</i> Message	Started by Source	Section 8.3.2.12.7.1	Section 8.3.3.14.1 , Section 8.3.3.14.2
Sink Gets Country Codes from a Source	1) <i>Get_Country_Codes</i> Message 2) <i>Country_Codes</i> Message	Started by Sink	Section 8.3.2.12.7.2	
VCONN Source Gets Country Codes from a Cable Plug	1) <i>Get_Country_Codes</i> Message 2) <i>Country_Codes</i> Message	Started by VCONN Source	Section 8.3.2.12.7.3	

8.3.2.1.3.19

AMS: Country Information

Table 8.23 “AMS: Country Information”

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
Source Gets Country Information from a Sink	1) <i>Get_Country_Info</i> Message 2) <i>Country_Info</i> Message	Started by Source	Section 8.3.2.12.8.1	Section 8.3.3.14.3 , Section 8.3.3.14.4
Sink Gets Country Information from a Source	1) <i>Get_Country_Info</i> Message 2) <i>Country_Info</i> Message	Started by Sink	Section 8.3.2.12.8.2	
VCONN Source Gets Country Information from a Cable Plug	1) <i>Get_Country_Info</i> Message 2) <i>Country_Info</i> Message	Started by VCONN Source	Section 8.3.2.12.8.3	

8.3.2.1.3.20

AMS: Revision Information

Table 8.24 “AMS: Revision Information”

AMS	Interruptible	Message Sequence	Conditions	AMS Ref
Source Gets Revision Information from a Sink	1) <i>Get_Revision</i> Message 2) <i>Revision</i> Message	Started by Source	Section 8.3.2.12.9.1	Section 8.3.3.15.1 , Section 8.3.3.15.2
Sink Gets Revision Information from a Source	1) <i>Get_Revision</i> Message 2) <i>Revision</i> Message	Started by Sink	Section 8.3.2.12.9.2	
VCONN Source Gets Revision Information from a Cable Plug	1) <i>Get_Revision</i> Message 2) <i>Revision</i> Message	Started by VCONN Source	Section 8.3.2.12.9.3	

8.3.2.1.3.21

AMS: Source Information

Table 8.25 “AMS: Source Information”

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
Sink Gets Source Information	1) <i>Get_Source_Cap_Extended</i> Message 2) <i>Source_Capabilities_Extended</i> Message	Started by Sink	Section 8.3.2.12.10.1	Section 8.3.3.10.1 Section 8.3.3.10.2
Dual-Role Source Gets Source Information from a Dual-Role Sink	1) <i>Get_Source_Cap_Extended</i> Message 2) <i>Source_Capabilities_Extended</i> Message	Started by Source	Section 8.3.2.12.10.2	Section 8.3.3.20.15 , Section 8.3.3.20.16

8.3.2.1.3.22

AMS: Security

Table 8.26 “AMS: Security”

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
Source requests security exchange with Sink	1) <i>Security_Request</i> Message	Started by Source	Section 8.3.2.13.1	Section 8.3.3.17.1 , Section 8.3.3.17.2 , Section 8.3.3.17.3
Sink requests security exchange with Source	1) <i>Security_Request</i> Message	Started by Sink	Section 8.3.2.13.2	
VCONN Source requests security exchange with Cable Plug	1) <i>Security_Request</i> Message	Started by VCONN Source	Section 8.3.2.13.3	
✖ Source responds to security exchange with Sink	1) <i>Security_Response</i> Message	Started by Source	Section 8.3.2.13.1	
✖ Sink responds to security exchange with Source	1) <i>Security_Response</i> Message	Started by Sink	Section 8.3.2.13.2	
VCONN Source requests security exchange with Cable Plug	1) <i>Security_Response</i> Message	Started by VCONN Source	Section 8.3.2.13.3	

8.3.2.1.3.23

AMS: Firmware Update

Table 8.27 “AMS: Firmware Update”

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
Source requests firmware update exchange with Sink	1) <i>Firmware_Update_Request</i> Message	Started by Source	Section 8.3.2.14.1	Section 8.3.3.18.1 , Section 8.3.3.18.2 , Section 8.3.3.18.3
Sink requests firmware update exchange with Source	1) <i>Firmware_Update_Request</i> Message	Started by Sink	Section 8.3.2.14.2	
✖ VCONN Source requests firmware update exchange with Cable Plug	1) <i>Firmware_Update_Request</i> Message	Started by VCONN Source	Section 8.3.2.14.3	
Source responds to firmware update exchange with Sink	1) <i>Firmware_Update_Response</i> Message	Started by Source	Section 8.3.2.14.1	
Sink responds to firmware update exchange with Source	1) <i>Firmware_Update_Response</i> Message	Started by Sink	Section 8.3.2.14.2	
✖ VCONN Source responds to firmware update exchange with Cable Plug	1) <i>Firmware_Update_Response</i> Message	Started by VCONN Source	Section 8.3.2.14.3	

8.3.2.1.3.24

AMS: Structured VDM

Table 8.28 "AMS: Structured VDM"

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
Initiator to Responder Discover Identity (ACK)	1) <i>Discover Identity</i> REQ Command 2) <i>Discover Identity</i> ACK Command	Started by Initiator	Section 8.3.2.15.1.1	Section 8.3.3.21.1 , Section 8.3.3.22.1
Initiator to Responder Discover Identity (NAK)	1) <i>Discover Identity</i> REQ Command 2) <i>Discover Identity</i> NAK Command		Section 8.3.2.15.1.2	
Initiator to Responder Discover Identity (BUSY)	1) <i>Discover Identity</i> REQ Command 2) <i>Discover Identity</i> BUSY Command		Section 8.3.2.15.1.3	
Initiator to Responder Discover SVIDs (ACK)	1) <i>Discover SVIDs</i> REQ Command 2) <i>Discover SVIDs</i> ACK Command		Section 8.3.2.15.2.1	Section 8.3.3.21.2 , Section 8.3.3.22.2
Initiator to Responder Discover SVIDs (NAK)	1) <i>Discover SVIDs</i> REQ Command 2) <i>Discover SVIDs</i> NAK Command		Section 8.3.2.15.2.2	
Initiator to Responder Discover SVIDs (BUSY)	1) <i>Discover SVIDs</i> REQ Command 2) <i>Discover SVIDs</i> BUSY Command		Section 8.3.2.15.2.3	
Initiator to Responder Discover Modes (ACK)	1) <i>Discover Modes</i> REQ Command 2) <i>Discover Modes</i> ACK Command		Section 8.3.2.15.3.1	Section 8.3.3.21.3 , Section 8.3.3.22.3
Initiator to Responder Discover Modes (NAK)	1) <i>Discover Modes</i> REQ Command 2) <i>Discover Modes</i> NAK Command		Section 8.3.2.15.3.2	
Initiator to Responder Discover Modes (BUSY)	1) <i>Discover Modes</i> REQ Command 2) <i>Discover Modes</i> BUSY Command		Section 8.3.2.15.3.3	
DFP to UFP Enter Mode	1) <i>Enter Mode</i> REQ Command 2) <i>Enter Mode</i> ACK Command	Started by DFP	Section 8.3.2.13.8	Section 8.3.3.23.1 , Section 8.3.3.24.1
DFP to UFP Exit Mode	1) <i>Exit Mode</i> REQ Command 2) <i>Exit Mode</i> ACK Command		Section 8.3.2.13.9	Section 8.3.3.23.2 , Section 8.3.3.24.2
DFP to Cable Plug Enter Mode	1) <i>Enter Mode</i> SEQ Command 2) <i>Enter Mode</i> ACK Command		Section 8.3.2.13.10	Section 8.3.3.23.2 , Section 8.3.3.25.4.1
DFP to Cable Plug Exit Mode	1) <i>Exit Mode</i> SEQ Command 2) <i>Exit Mode</i> ACK Command		Section 8.3.2.13.11	Section 8.3.3.23.2 , Section 8.3.3.25.4.2
Initiator to Responder Attention	1) <i>Attention</i> REQ Command	Started by Initiator	Section 8.3.2.13.12	Section 8.3.3.21.4 , Section 8.3.3.22.4

8.3.2.1.3.25

AMS: Built-In Self-Test (BIST)

Table 8.29 “AMS: Built-In Self-Test (BIST)”

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
BIST Carrier Mode	1) <i>BIST (BIST Carrier Mode)</i> Message	Started by Tester	Section 8.3.2.16.1	Section 8.3.3.27.1
✖ BIST Test Data	1) <i>BIST (BIST Test Data)</i> Message		Section 8.3.2.16.2	Section 8.3.3.27.3
✖ BIST Shared Capacity Test Mode	1) <i>BIST (BIST Shared Test Mode Entry)</i> Message 2) Series of Messages 3) <i>BIST (BIST Shared Test Mode Exit)</i> Message		Section 8.3.2.16.3	Section 8.3.3.27.3

8.3.2.1.3.26

AMS: Enter USB

Table 8.30 “AMS: Enter USB”

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
UFP Entering USB4® Mode (Accept)	1) <i>Enter_USB</i> Message 2) <i>Accept</i> Message	Started by DFP	Section 8.3.2.17.1.1	Section 8.3.3.16.1 , Section 8.3.3.16.2
✖ UFP Entering USB4® Mode (Reject)	1) <i>Enter_USB</i> Message 2) <i>Reject</i> Message		Section 8.3.2.17.1.2	
UFP Entering USB4® Mode (Wait)	1) <i>Enter_USB</i> Message 2) <i>Wait</i> Message		Section 8.3.2.17.1.3	
Cable Plug Entering USB4® Mode (Accept)	1) <i>Enter_USB</i> Message 2) <i>Accept</i> Message		Section 8.3.2.17.2.1	
Cable Plug Entering USB4® Mode (Reject)	1) <i>Enter_USB</i> Message 2) <i>Reject</i> Message		Section 8.3.2.17.2.2	
Cable Plug Entering USB4® Mode (Wait)	1) <i>Enter_USB</i> Message 2) <i>Wait</i> Message		Section 8.3.2.17.2.3	

8.3.2.1.3.27

AMS: Unstructured VDM

Table 8.31 “AMS: Unstructured VDM”

AMS	Message Sequence	Conditions	AMS Ref	State Machine Ref
Unstructured VDM	1) Unstructured <i>Vendor Defined</i> Message	Section 8.3.2.18.1	Section 8.3.2.18.1	
VDEM	1) <i>Vendor Defined Extended</i> Message	Section 8.3.2.18.2	Section 8.3.2.18.2	

8.3.2.1.3.28

AMS: Hard Reset

Table 8.32 “AMS: Hard Reset”

AMS	Interruptible	Message Sequence	Conditions	AMS Ref	State Machine Ref
Source Initiated Hard Reset	No	1) Hard Reset Signaling 2) SPR Explicit Contract Negotiation AMS	Started by Source	Section 8.3.2.5.1 Section 8.3.2.2.1.1	Section 8.3.3.2, Section 8.3.3.3
Sink Initiated Hard Reset	No	1) Hard Reset Signaling 2) SPR Explicit Contract Negotiation AMS	Started by Sink	Section 8.3.2.5.2 Section 8.3.2.2.1.1	
Source Initiated Hard Reset – Sink Long Reset	No	1) Hard Reset Signaling 2) SPR Explicit Contract Negotiation AMS	Started by Source	Section 8.3.2.5.3 Section 8.3.2.2.1.1	

8.3.2.2 Power Negotiation

8.3.2.2.1 SPR

8.3.2.2.1.1 SPR Explicit Contract Negotiation

8.3.2.2.1.1.1 SPR Explicit Contract Negotiation (Accept)

Figure 8-5 “Successful Fixed, Variable or Battery SPR Power Negotiation” illustrates an example of a successful Message flow while negotiating an Explicit Contract in SPR Mode. The negotiation goes through 5 distinct phases:

- The Source sends out its power capabilities in a **Source_Capabilities** Message.
- The Sink evaluates these capabilities, and, in the request, phase selects one power level by sending a **Request** Message.
- The Source evaluates the request and accepts the request with an **Accept** Message.
- The Source transitions to the new power level and then informs the Sink by sending a **PS_RDY** Message.
- The Sink starts using the new power level.
- For SPR PPS operation:
 - the Source starts its keep alive timer.
 - the Sink starts its request timer to send periodic **Request** Messages.

Figure 8-5 “Successful Fixed, Variable or Battery SPR Power Negotiation”

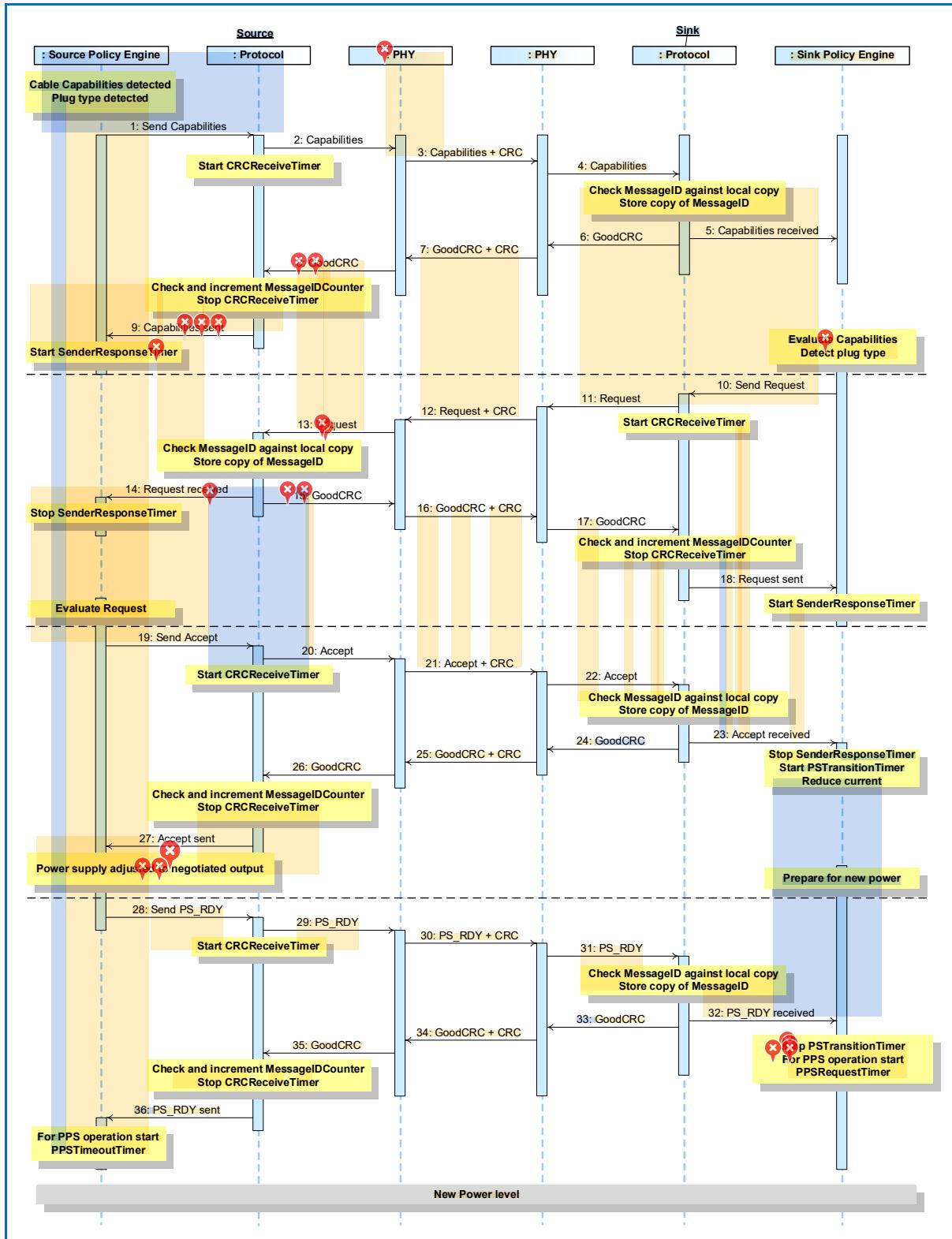


Table 8.33 “Steps for a successful Power Negotiation” below provides a detailed explanation of what happens at each labeled step in **Figure 8-5 “Successful Fixed, Variable or Battery SPR Power Negotiation”** above.

Table 8.33 “Steps for a successful Power Negotiation”

Step	Source	Sink
1	The Cable Capabilities or Plug Type are detected if these are not already known (see Section 4.4 “Cable Type Detection”). Policy Engine directs the Protocol Layer to send a Source_Capabilities Message that represents the power supply's present capabilities.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Source_Capabilities Message. Starts CRCReceiveTimer .	Physical Layer receives the Source_Capabilities Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Source_Capabilities Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Source_Capabilities Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Source_Capabilities Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the Source_Capabilities Message sent by the Source, detects the plug type if this is necessary (see Section 4.4 “Cable Type Detection”) and selects which power it would like. It tells the Protocol Layer to form the data (e.g., Power Data Object) that represents its Request into a Message.
11		Protocol Layer creates the Request Message and passes to Physical Layer.
12	Physical Layer receives the Request Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Request Message. Starts CRCReceiveTimer .
13	Physical Layer removes the CRC and forwards the Request Message to the Protocol Layer.	

14	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine. Policy Engine stops <i>SenderResponseTimer</i> .	
15	The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.	
16	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		The protocol Layer verifies and increments the <i>MessageIDCounter</i> . It informs the Policy Engine that the <i>Request</i> Message was successfully sent. The Protocol Layer stops the <i>CRCReceiveTimer</i> . The Policy Engine starts <i>SenderResponseTimer</i> .
19	Policy Engine evaluates the <i>Request</i> Message sent by the Sink and decides if it can meet the request. It tells the Protocol Layer to form an <i>Accept</i> Message.	
20	The Protocol Layer forms the <i>Accept</i> Message that is passed to the Physical Layer.	
21	Physical Layer appends CRC and sends the <i>Accept</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer forwards the <i>Accept</i> Message to the Protocol Layer.
23		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that an <i>Accept</i> Message has been received. The Policy Engine stops <i>SenderResponseTimer</i> , starts the <i>PSTransitionTimer</i> and reduces its current draw. The Device Policy Manager prepares the Power supply for transition to the new power level.
24		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
25	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Message.
26	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops the <i>CRCReceiveTimer</i> .	
27	The Protocol Layer informs the Policy Engine that an <i>Accept</i> Message was successfully sent.	
Power supply Adjusts its Output to the Negotiated Value		

28	The Device Policy Manager informs the Policy Engine that the power supply has settled at the new operating condition and tells the Protocol Layer to send a <i>PS_RDY</i> Message.	
29	The Protocol Layer forms the <i>PS_RDY</i> Message.	
30	Physical Layer appends CRC and sends the <i>PS_RDY</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the <i>PS_RDY</i> Message and compares the CRC it calculated with the one sent to verify the Message.
31		Physical Layer forwards the <i>PS_RDY</i> Message to the Protocol Layer.
32		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a RS_RDY has been received. The Policy Engine stops the <i>PSTransitionTimer</i> . When in SPR PPS operation the Policy Engine starts the <i>SinkPPSPeriodicTimer</i> .
33		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
34	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Message.
35	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> . Stops the <i>CRCReceiveTimer</i> .	
36	The Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent.	
37	When in SPR PPS operation the Policy Engine starts the <i>SourcePPSCommTimer</i> .	
New Power Level Negotiated		

8.3.2.2.1.1.2

SPR Explicit Contract Negotiation (Reject)

Figure 8-6 “Rejected Fixed, Variable or Battery SPR Power Negotiation” illustrates an example of a Message flow where the request is rejected while negotiating an Explicit Contract in SPR Mode. The negotiation goes through the following phases:

- The Source sends out its power capabilities in a **Source_Capabilities** Message.
- The Sink evaluates these capabilities, and, in the request, phase selects one power level by sending a **Request** Message.
- The Source evaluates the request and rejects the request with a **Reject** Message.

Figure 8-6 “Rejected Fixed, Variable or Battery SPR Power Negotiation”

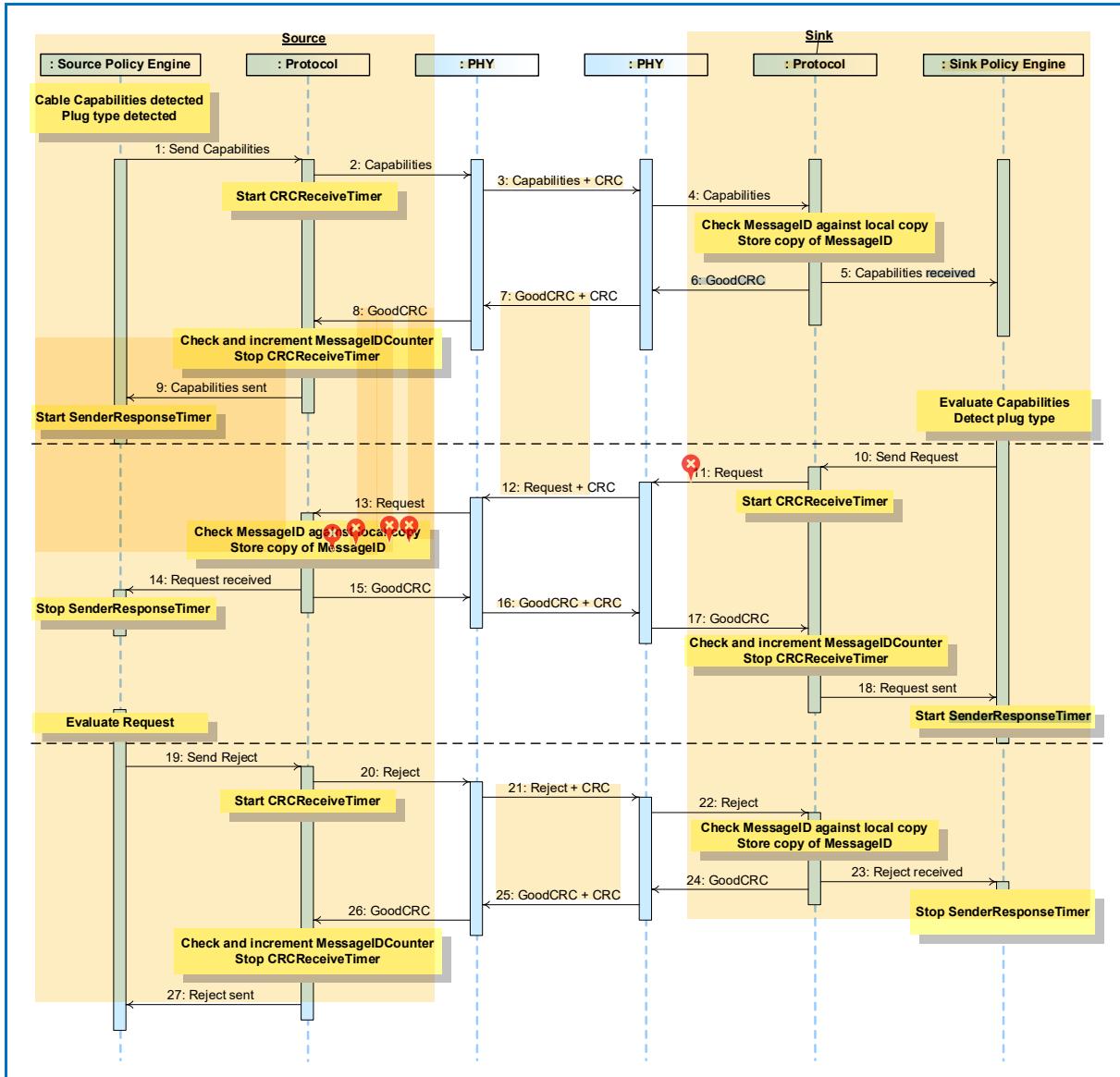


Table 8.34 “Steps for a rejected Power Negotiation” below provides a detailed explanation of what happens at each labeled step in **Figure 8-6 “Rejected Fixed, Variable or Battery SPR Power Negotiation”** above.

Table 8.34 “Steps for a rejected Power Negotiation”

Step	Source	Sink
1	The Cable Capabilities or Plug Type are detected if these are not already known (see Section 4.4 “Cable Type Detection”). Policy Engine directs the Protocol Layer to send a Source_Capabilities Message that represents the power supply's present capabilities.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Source_Capabilities Message. Starts CRCReceiveTimer .	Physical Layer receives the Source_Capabilities Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Source_Capabilities Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Source_Capabilities Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Source_Capabilities Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the Source_Capabilities Message sent by the Source, detects the plug type if this is necessary (see Section 4.4 “Cable Type Detection”) and selects which power it would like. It tells the Protocol Layer to form the data (e.g., Power Data Object) that represents its Request into a Message.
11		Protocol Layer creates the Request Message and passes to Physical Layer.
12	Physical Layer receives the Request Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Request Message. Starts CRCReceiveTimer .
13	Physical Layer removes the CRC and forwards the Request Message to the Protocol Layer.	

14	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine. Policy Engine stops SenderResponseTimer .	
15	The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.	
16	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer forwards the GoodCRC Message to the Protocol Layer.
18		The protocol Layer verifies and increments the MessageIDCounter . It informs the Policy Engine that the Request Message was successfully sent. The Protocol Layer stops the CRCReceiveTimer . The Policy Engine starts SenderResponseTimer .
19	Policy Engine evaluates the Request Message sent by the Sink and decides if it can meet the request. It tells the Protocol Layer to form a Reject Message.	
20	The Protocol Layer forms the Reject Message that is passed to the Physical Layer.	
21	Physical Layer appends CRC and sends the Reject Message. Starts CRCReceiveTimer .	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer forwards the Reject Message to the Protocol Layer.
23		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a Reject Message has been received. The Policy Engine stops SenderResponseTimer .
24		The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.
25	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Message.
26	Physical Layer forwards the GoodCRC Message to the Protocol Layer. The Protocol Layer verifies and increments the MessageIDCounter and stops the CRCReceiveTimer .	
27	The Protocol Layer informs the Policy Engine that a Reject Message was successfully sent.	

8.3.2.2.1.1.3

SPR Explicit Contract Negotiation (Wait)

Figure 8-7 “Wait response to Fixed, Variable or Battery SPR Power Negotiation” illustrates an example of a Message flow where the request is responded to with wait while negotiating an Explicit Contract in SPR Mode. The negotiation goes through the following phases:

- The Source sends out its power capabilities in a **Source_Capabilities** Message.
- The Sink evaluates these capabilities, and, in the request, phase selects one power level by sending a **Request** Message.
- The Source evaluates the request and rejects the request with a **Wait** Message.

Figure 8-7 “Wait response to Fixed, Variable or Battery SPR Power Negotiation”

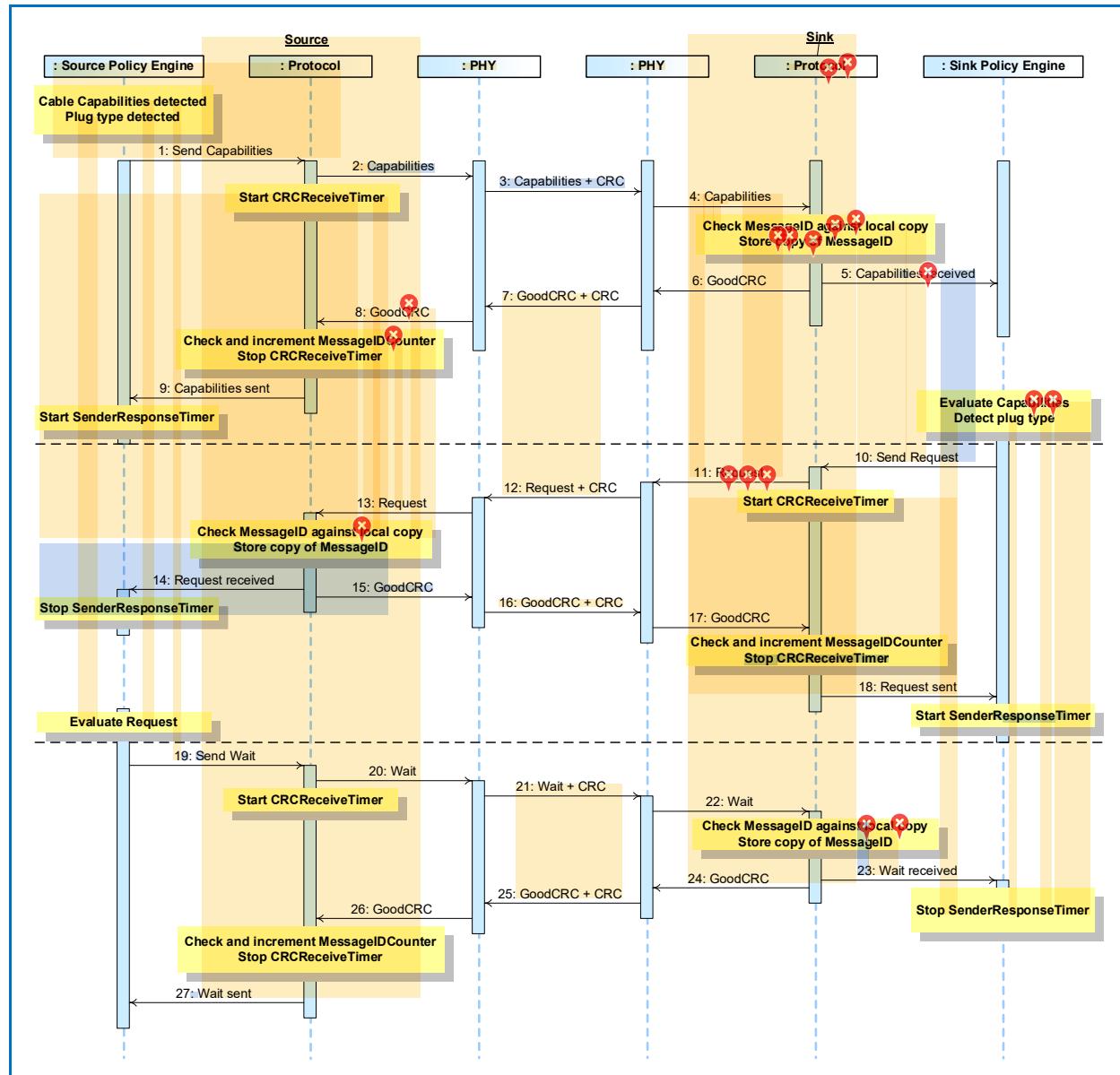


Table 8.35 “Steps for a Wait response to a Power Negotiation” below provides a detailed explanation of what happens at each labeled step in **Figure 8-7 “Wait response to Fixed, Variable or Battery SPR Power Negotiation”** above.

Table 8.35 “Steps for a Wait response to a Power Negotiation”

Step	Source	Sink
1	The Cable Capabilities or Plug Type are detected if these are not already known (see Section 4.4 “Cable Type Detection”). Policy Engine directs the Protocol Layer to send a Source_Capabilities Message that represents the power supply’s present capabilities.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Source_Capabilities Message. Starts CRCReceiveTimer .	Physical Layer receives the Source_Capabilities Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Source_Capabilities Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Source_Capabilities Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Source_Capabilities Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the Source_Capabilities Message sent by the Source, detects the plug type if this is necessary (see Section 4.4 “Cable Type Detection”) and selects which power it would like. It tells the Protocol Layer to form the data (e.g., Power Data Object) that represents its Request into a Message.
11		Protocol Layer creates the Request Message and passes to Physical Layer.
12	Physical Layer receives the Request Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Request Message. Starts CRCReceiveTimer .
13	Physical Layer removes the CRC and forwards the Request Message to the Protocol Layer.	

14	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine. Policy Engine stops <i>SenderResponseTimer</i> .	
15	The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.	
16	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		The protocol Layer verifies and increments the <i>MessageIDCounter</i> . It informs the Policy Engine that the <i>Request</i> Message was successfully sent. The Protocol Layer stops the <i>CRCReceiveTimer</i> . The Policy Engine starts <i>SenderResponseTimer</i> .
19	Policy Engine evaluates the <i>Request</i> Message sent by the Sink and decides if it can meet the request. It tells the Protocol Layer to form a <i>Wait</i> Message.	
20	The Protocol Layer forms the <i>Wait</i> Message that is passed to the Physical Layer.	
21	Physical Layer appends CRC and sends the <i>Wait</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer forwards the <i>Wait</i> Message to the Protocol Layer.
23		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a <i>Wait</i> Message has been received. The Policy Engine stops <i>SenderResponseTimer</i> .
24		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
25	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message. ✗	Physical Layer appends CRC and sends the Message.
26	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops the <i>CRCReceiveTimer</i> .	
27	The Protocol Layer informs the Policy Engine that a <i>Wait</i> Message was successfully sent.	

8.3.2.2.1.2

Reclaiming Power with GotoMin Message

This is an example of a GotoMin operation. [Figure 8-8 “Successful GotoMin operation”](#) shows the Messages as they flow across the bus and within the devices to accomplish the GotoMin.

[Figure 8-8 “Successful GotoMin operation”](#)

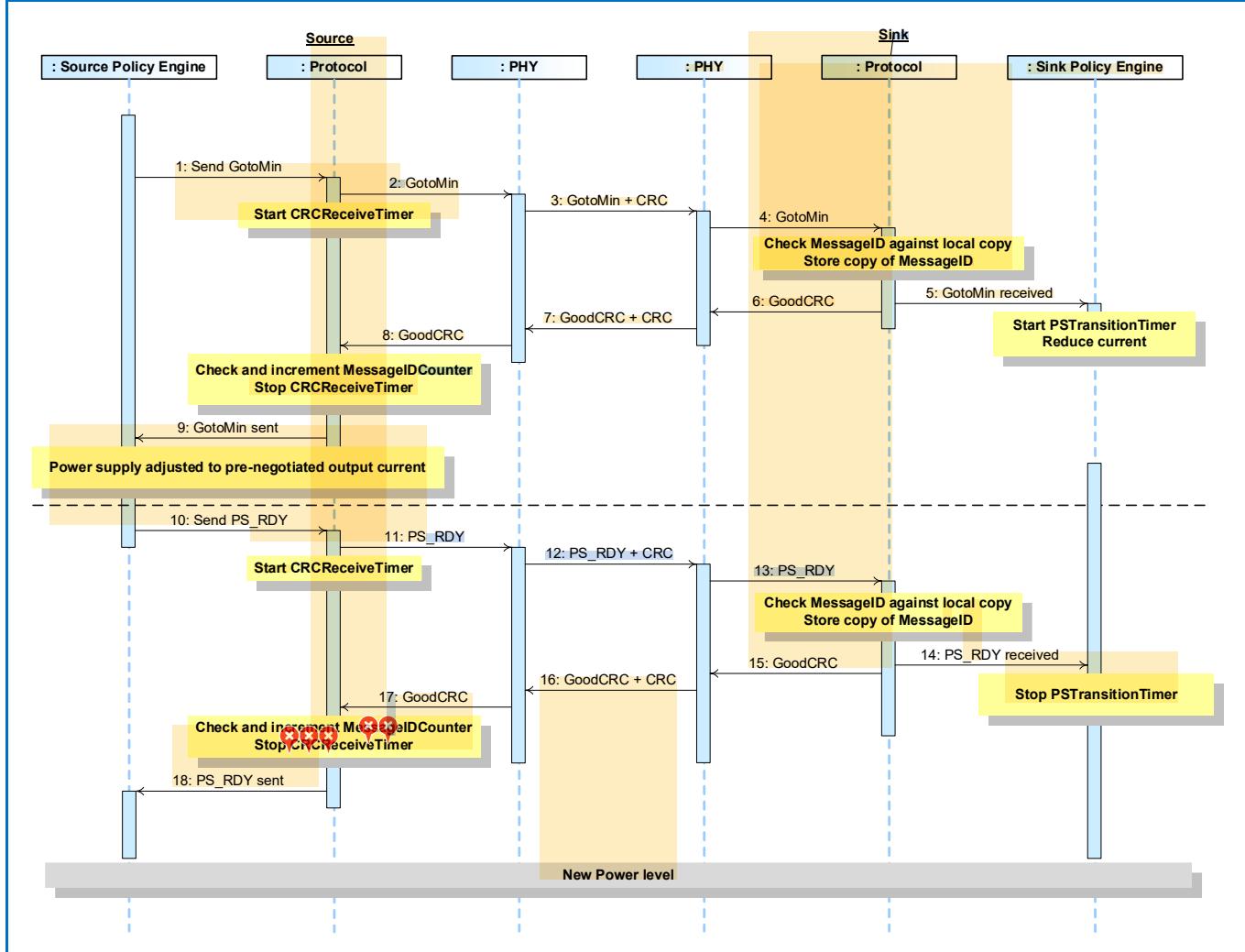


Table 8.36 “Steps for a GotoMin Negotiation” provides a detailed explanation of what happens at each labeled step in **Figure 8-8 “Successful GotoMin operation”** above.

Table 8.36 “Steps for a GotoMin Negotiation”

Step	Source	Sink
1	Policy Engine tells the Protocol Layer to form a GotoMin Message.	
2	The Protocol Layer forms the GotoMin Message that is passed to the Physical Layer.	
3	Physical Layer appends CRC and sends the GotoMin Message. Starts CRCReceiveTimer .	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer forwards the GotoMin Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a GotoMin Message has been received. The Policy starts the PSTransitionTimer and reduces its current draw. The Policy Engine prepares the Power supply for transition to the new power level.
6		The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.
7	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Message.
8	Physical Layer forwards the GoodCRC Message to the Protocol Layer. The Protocol Layer verifies and increments the MessageIDCounter and stops the CRCReceiveTimer .	
9	The Protocol Layer informs the Policy Engine that a GotoMin Message was successfully sent.	
Power supply Adjusts its Output to the Negotiated Value		
10	Policy Engine sees the power supply has settled at the new operating condition and tells the Protocol Layer to send a PS_RDY Message.	
11	The Protocol Layer forms the PS_RDY Message.	
12	Physical Layer appends CRC and sends the PS_RDY Message. Starts CRCReceiveTimer .	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
13		Physical Layer forwards the PS_RDY Message to the Protocol Layer.

14		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a <i>PS_RDY</i> Message has been received. The Policy Engine stops the <i>PSTransitionTimer</i> .
15		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
16	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Message.
17	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops the <i>CRCReceiveTimer</i> .	
18	The Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent.	

New Power Level Negotiated

8.3.2.2.1.3

SPR PPS Keep Alive

This is an example of SPR PPS keep alive operation during an Explicit Contract with SPR PPS as the APDO. [Figure 8-9 “SPR PPS Keep Alive”](#) shows the Messages as they flow across the bus and within the devices to accomplish the keep alive.

Figure 8-9 “SPR PPS Keep Alive”

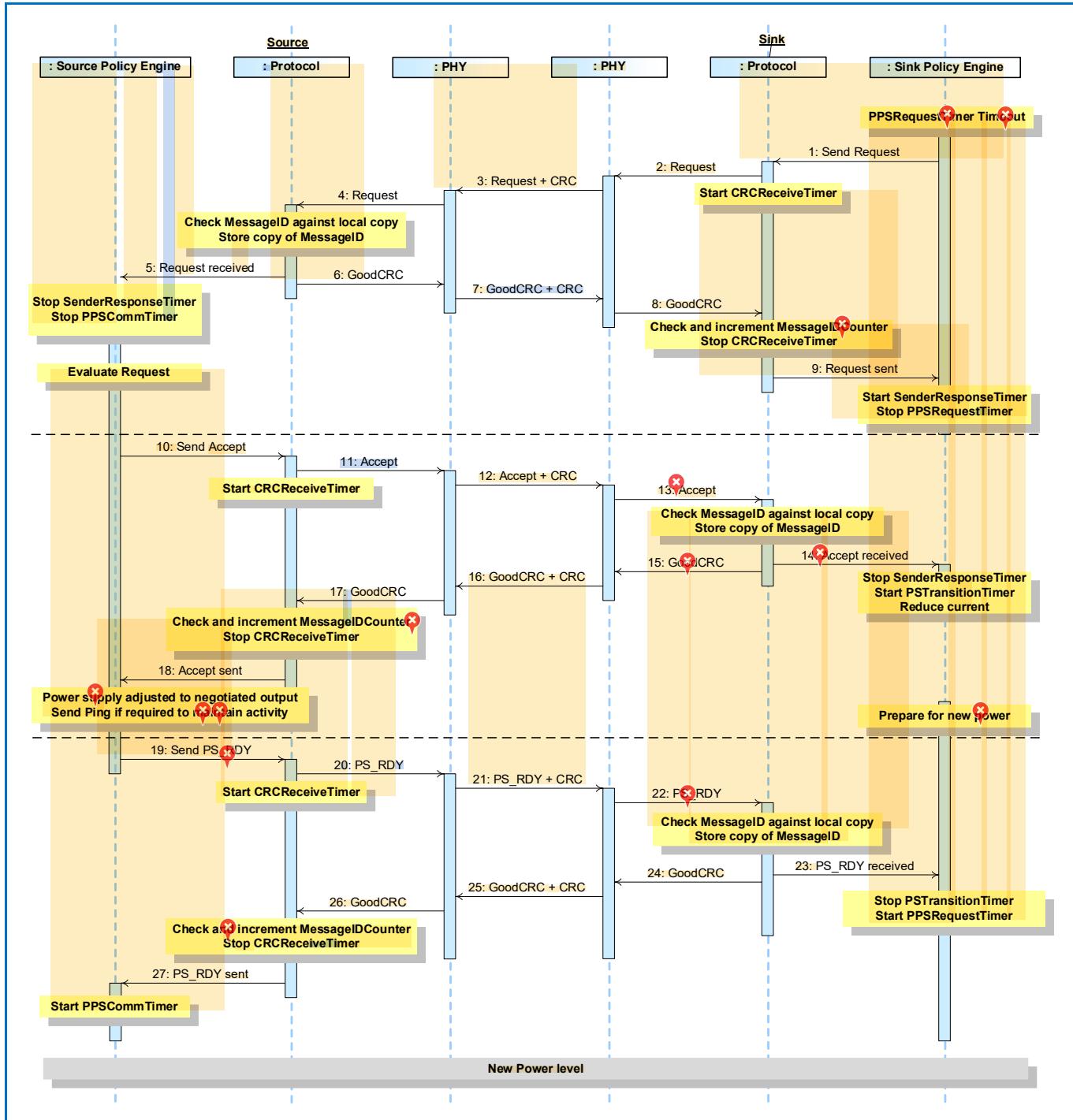


Table 8.37 “Steps for SPR PPS Keep Alive” below provides a detailed explanation of what happens at each labeled step in **Figure 8-9 “SPR PPS Keep Alive”** above.

Table 8.37 “Steps for SPR PPS Keep Alive”

Step	Source	Sink
1		The <i>SinkPPSPeriodicTimer</i> times out in the Policy Engine. The Policy Engine tells the Protocol Layer to form a <i>Request</i> Message. The Protocol Layer creates the <i>Request</i> Message and passes it to Physical Layer.
2	Physical Layer receives the <i>Request</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Request</i> Message. Starts <i>CRCReceiveTimer</i> .
3	Physical Layer removes the CRC and forwards the <i>Request</i> Message to the Protocol Layer.	
4	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine. Policy Engine stops the <i>SourcePPSCommTimer</i> .	
5	The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.	
6	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives the <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
7		Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer.
8		The protocol Layer verifies and increments the <i>MessageIDCounter</i> . It informs the Policy Engine that the <i>Request</i> Message was successfully sent. The Protocol Layer stops the <i>CRCReceiveTimer</i> . The Policy Engine starts <i>SenderResponseTimer</i> .
9	Policy Engine requests the Device Policy Manager to evaluate the <i>Request</i> Message sent by the Sink and decides if the Source can meet the request. The Policy Engine tells the Protocol Layer to form an <i>Accept</i> Message.	
10	The Protocol Layer forms the <i>Accept</i> Message that is passed to the Physical Layer.	
11	Physical Layer appends CRC and sends the <i>Accept</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the <i>Accept</i> Message and compares the CRC it calculated with the one sent to verify the Message.
12		Physical Layer forwards the <i>Accept</i> Message to the Protocol Layer. 

13		<p>Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value.</p> <p>Protocol Layer informs the Policy Engine that an Accept Message has been received. The Policy Engine stops SenderResponseTimer, starts the PSTransitionTimer and reduces its current draw.</p> <p>The Device Policy Manager prepares the Power supply for transition to the new power level.</p>
14		The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.
15	Physical Layer receives the GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.✖	Physical Layer appends CRC and sends the GoodCRC Message.
16	Physical Layer forwards the GoodCRC Message to the Protocol Layer. The Protocol Layer verifies and increments the MessageIDCounter and stops the CRCReceiveTimer .	
17	The Protocol Layer informs the Policy Engine that an Accept Message was successfully sent.	
Power supply Adjusts its Output to the Negotiated Value		
18	The Device Policy Manager informs the Policy Engine that the power supply has settled at the new operating condition and tells the Protocol Layer to send a PS_RDY Message.	
19	The Protocol Layer forms the PS_RDY Message.	
20	Physical Layer appends CRC and sends the PS_RDY Message. Starts CRCReceiveTimer .✖	Physical Layer receives the PS_RDY Message and compares the CRC it calculated with the one sent to verify the Message.
21		Physical Layer forwards the PS_RDY Message to the Protocol Layer.
22		<p>Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value.</p> <p>Protocol Layer informs the Policy Engine that a RS_RDY has been received. The Policy Engine stops the PSTransitionTimer.</p> <p>When in SPR PPS operation the Policy Engine starts the SinkPPSPeriodicTimer.</p>
23		The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.
24	Physical Layer receives the GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.✖	Physical Layer appends CRC and sends the GoodCRC Message.
25	Physical Layer forwards the GoodCRC Message to the Protocol Layer. The Protocol Layer verifies and increments the MessageIDCounter . Stops the CRCReceiveTimer .	
26	The Protocol Layer informs the Policy Engine that the PS_RDY Message was successfully sent.✖	

27	When in SPR PPS operation the Policy Engine starts the <i>SourcePPSCommTimer</i> .	
----	--	--

8.3.2.2.1.4

SPR Sink Makes Request

8.3.2.2.1.4.1

SPR Sink Makes Request (Accept)

This is an example of SPR when a Sink makes a Request which is Accepted during an Explicit Contract. **Figure 8-10 “SPR Sink Makes Request (Accept)”** shows the Messages as they flow across the bus and within the devices to accomplish the keep alive.

 **Figure 8-10 “SPR Sink Makes Request (Accept)”**

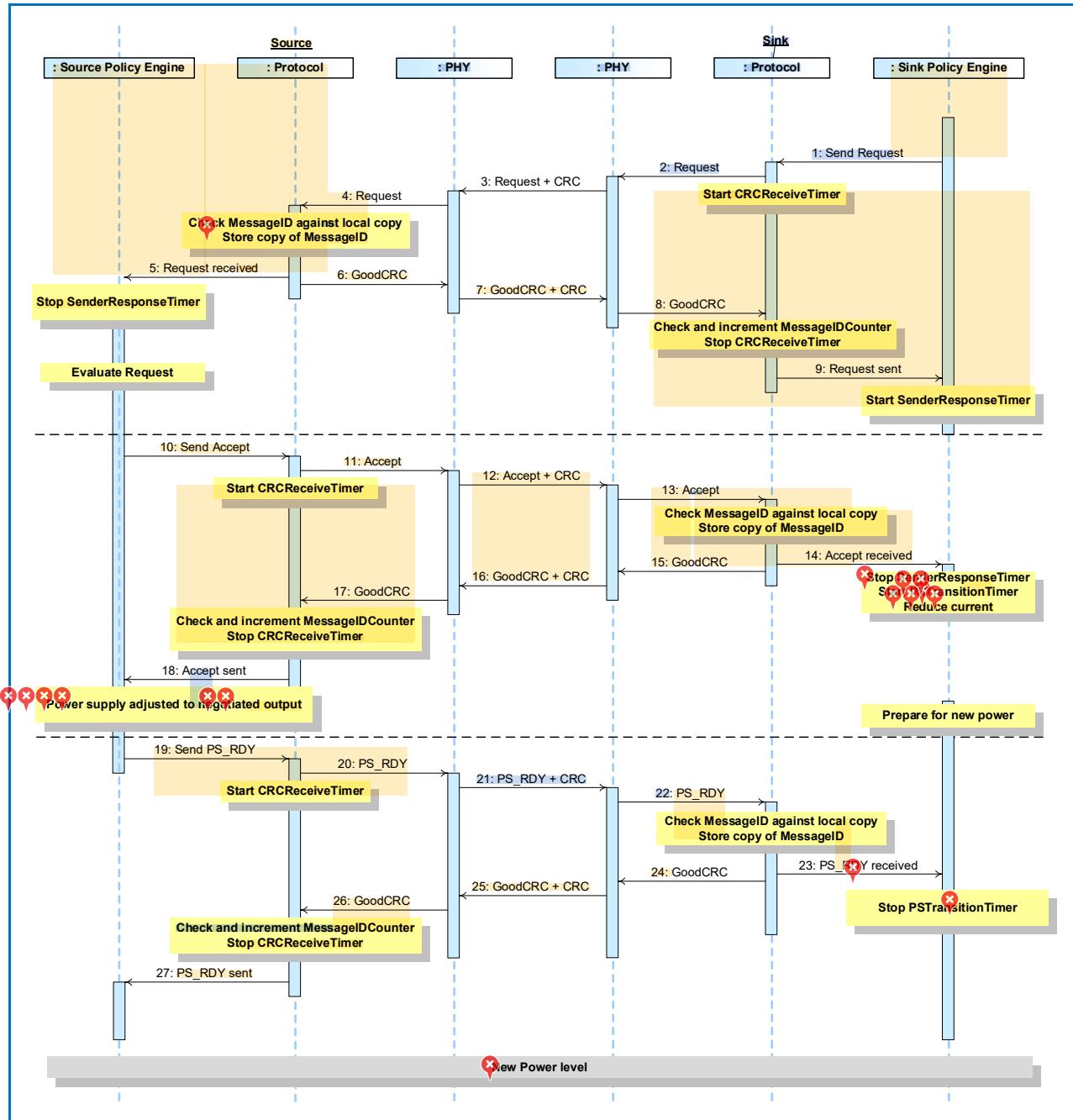


Table 8.38 “Steps for SPR Sink Makes Request (Accept)” below provides a detailed explanation of what happens at each labeled step in **Figure 8-10 “SPR Sink Makes Request (Accept)”** above.

Table 8.38 “Steps for SPR Sink Makes Request (Accept)”

Step	Source	Sink
1		DPM tells the Policy Engine to request a different power level. The Policy Engine tells the Protocol Layer to form a Request Message. The Protocol Layer creates the Request Message and passes it to Physical Layer.
2	Physical Layer receives the Request Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Request Message. Starts CRCReceiveTimer .
3	Physical Layer removes the CRC and forwards the Request Message to the Protocol Layer.	
4	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine.	
5	The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.	
6	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
7		Physical Layer forwards the GoodCRC Message to the Protocol Layer.
8		The protocol Layer verifies and increments the MessageIDCounter . It informs the Policy Engine that the Request Message was successfully sent. The Protocol Layer stops the CRCReceiveTimer . The Policy Engine starts SenderResponseTimer .
9	Policy Engine requests the Device Policy Manager to evaluate the Request Message sent by the Sink and decides if the Source can meet the request. The Policy Engine tells the Protocol Layer to form an Accept Message.	
10	The Protocol Layer forms the Accept Message that is passed to the Physical Layer.	
11	Physical Layer appends CRC and sends the Accept Message. Starts CRCReceiveTimer .	Physical Layer receives the Accept Message and compares the CRC it calculated with the one sent to verify the Message.
12		Physical Layer forwards the Accept Message to the Protocol Layer.

13		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that an Accept Message has been received. The Policy Engine stops SenderResponseTimer , starts the PSTransitionTimer and reduces its current draw. The Device Policy Manager prepares the Power supply for transition to the new power level.
14		The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.
15	Physical Layer receives the GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
16	Physical Layer forwards the GoodCRC Message to the Protocol Layer. The Protocol Layer verifies and increments the MessageIDCounter and stops the CRCReceiveTimer .	
17	The Protocol Layer informs the Policy Engine that an Accept Message was successfully sent.	
Power supply Adjusts its Output to the Negotiated Value		
18	The Device Policy Manager informs the Policy Engine that the power supply has settled at the new operating condition and tells the Protocol Layer to send a PS_RDY Message.	
19	The Protocol Layer forms the PS_RDY Message.	
20	Physical Layer appends CRC and sends the PS_RDY Message. Starts CRCReceiveTimer .	Physical Layer receives the PS_RDY Message and compares the CRC it calculated with the one sent to verify the Message.
21		Physical Layer forwards the PS_RDY Message to the Protocol Layer.
22		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a RS_RDY has been received. The Policy Engine stops the PSTransitionTimer .
23		The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.
24	Physical Layer receives the GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
25	Physical Layer forwards the GoodCRC Message to the Protocol Layer. The Protocol Layer verifies and increments the MessageIDCounter . Stops the CRCReceiveTimer .	
26	The Protocol Layer informs the Policy Engine that the PS_RDY Message was successfully sent.	
New Power Level Negotiated		

8.3.2.2.1.4.2

SPR Sink Makes Request (Reject)

This is an example of SPR when a Sink makes a Request which is Rejected during an Explicit Contract. [Figure 8-11 “SPR Sink Makes Request \(Reject\)”](#) shows the Messages as they flow across the bus and within the devices to accomplish the keep alive.

[Figure 8-11 “SPR Sink Makes Request \(Reject\)”](#)

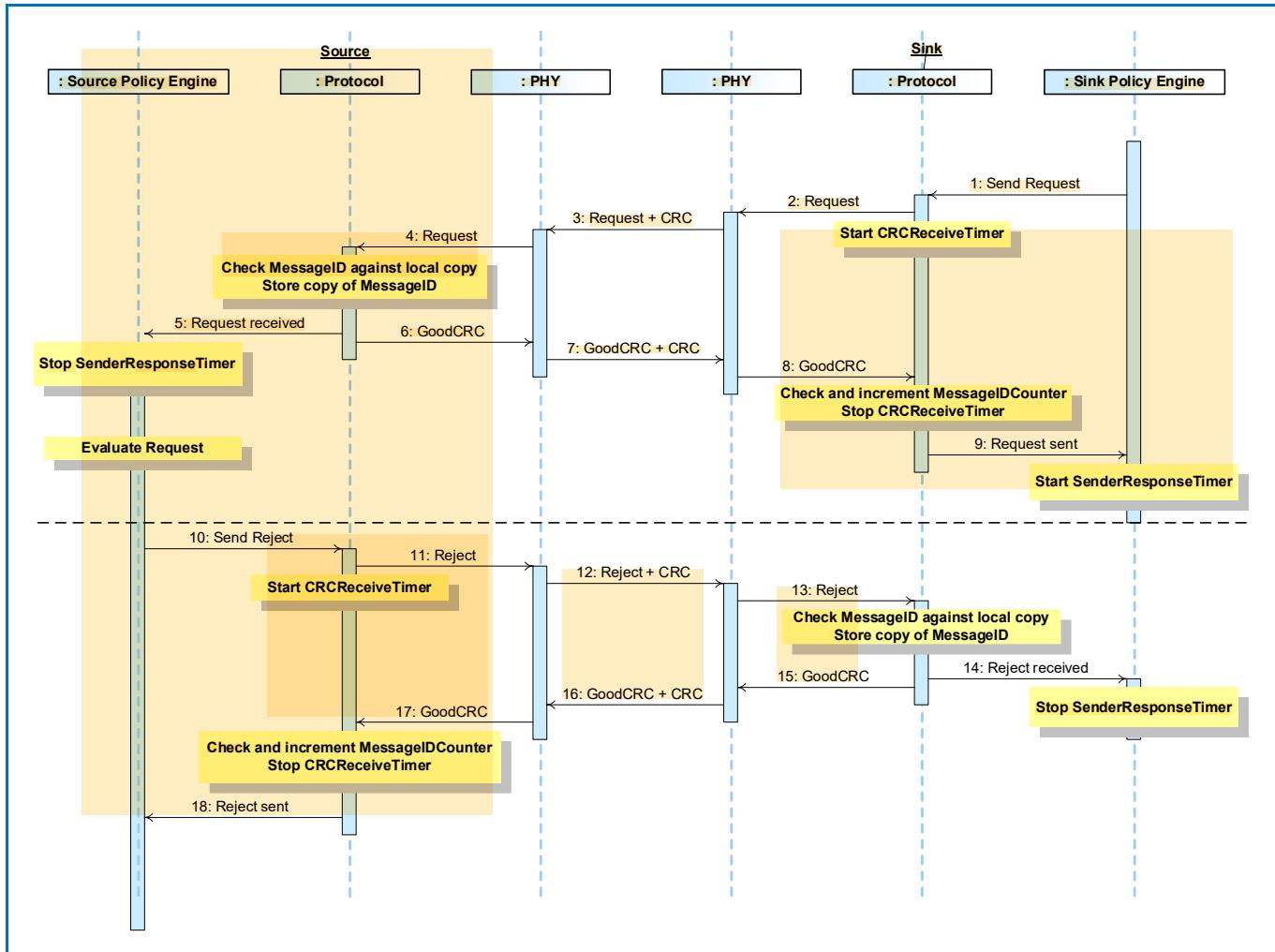


Table 8.39 “Steps for SPR Sink Makes Request (Reject)” below provides a detailed explanation of what happens at each labeled step in **Figure 8-11 “SPR Sink Makes Request (Reject)”** above.

Table 8.39 “Steps for SPR Sink Makes Request (Reject)”

Step	Source	Sink
1		DPM tells the Policy Engine to request a different power level. The Policy Engine tells the Protocol Layer to form a Request Message. The Protocol Layer creates the Request Message and passes it to Physical Layer.
2	Physical Layer receives the Request Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Request Message. Starts CRCReceiveTimer .
3	Physical Layer removes the CRC and forwards the Request Message to the Protocol Layer.	
4	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine.	
5	The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.	
6	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
7		Physical Layer forwards the GoodCRC Message to the Protocol Layer.
8		The protocol Layer verifies and increments the MessageIDCounter . It informs the Policy Engine that the Request Message was successfully sent. The Protocol Layer stops the CRCReceiveTimer . The Policy Engine starts SenderResponseTimer .
9	Policy Engine requests the Device Policy Manager to evaluate the Request Message sent by the Sink and decides if the Source can meet the request. The Policy Engine tells the Protocol Layer to form a Reject Message.	
10	The Protocol Layer forms the Reject Message that is passed to the Physical Layer.	
11	Physical Layer appends CRC and sends the Reject Message. Starts CRCReceiveTimer .	Physical Layer receives the Reject Message and compares the CRC it calculated with the one sent to verify the Message.
12		Physical Layer forwards the Reject Message to the Protocol Layer.

13		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that an <i>Reject</i> Message has been received. The Policy Engine informs the Device Policy Manager that the Request has been rejected.
14		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
15	Physical Layer receives the <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
16	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops the <i>CRCReceiveTimer</i> .	
17	The Protocol Layer informs the Policy Engine that a <i>Reject</i> Message was successfully sent.	

8.3.2.2.1.4.3

SPR Sink Makes Request (Wait)

This is an example of SPR when a Sink makes a Request which is responded to with Wait during an Explicit Contract. **Figure 8-12 “SPR Sink Makes Request (Wait)”** shows the Messages as they flow across the bus and within the devices to accomplish the keep alive.

Figure 8-12 “SPR Sink Makes Request (Wait)”

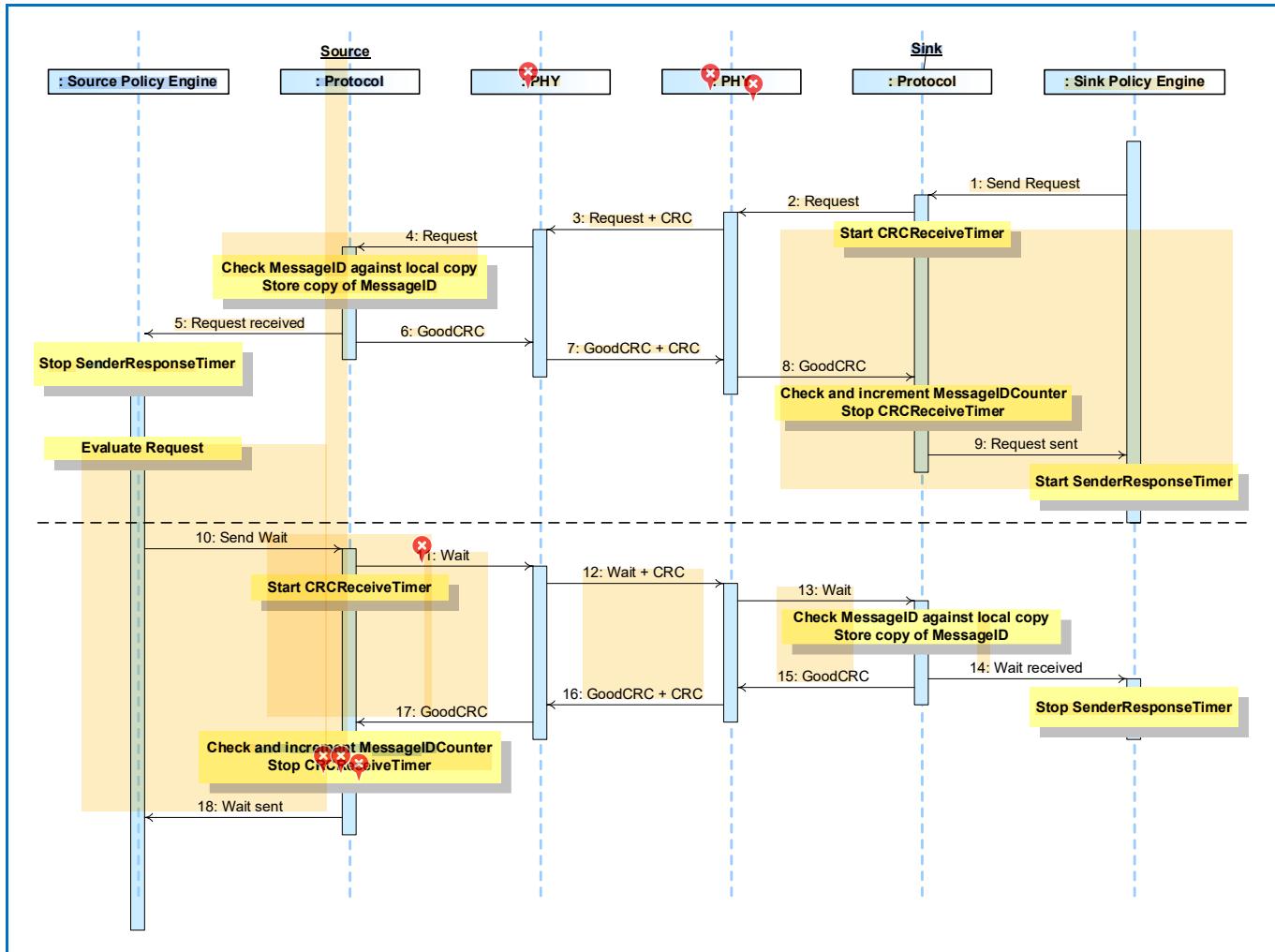


Table 8.40 “Steps for SPR Sink Makes Request (Wait)” below provides a detailed explanation of what happens at each labeled step in **Figure 8-12 “SPR Sink Makes Request (Wait)”** above.

Table 8.40 “Steps for SPR Sink Makes Request (Wait)”

Step	Source	Sink
1		DPM tells the Policy Engine to request a different power level. The Policy Engine tells the Protocol Layer to form a Request Message. The Protocol Layer creates the Request Message and passes it to Physical Layer.
2	Physical Layer receives the Request Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Request Message. Starts CRCReceiveTimer .
3	Physical Layer removes the CRC and forwards the Request Message to the Protocol Layer.	
4	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine.	
5	The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.	
6	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
7		Physical Layer forwards the GoodCRC Message to the Protocol Layer.
8		The protocol Layer verifies and increments the MessageIDCounter . It informs the Policy Engine that the Request Message was successfully sent. The Protocol Layer stops the CRCReceiveTimer . The Policy Engine starts SenderResponseTimer .
9	Policy Engine requests the Device Policy Manager to evaluate the Request Message sent by the Sink and decides if the Source can meet the request. The Policy Engine tells the Protocol Layer to form a Wait Message.	
10	The Protocol Layer forms the Wait Message that is passed to the Physical Layer.	
11	Physical Layer appends CRC and sends the Wait Message. Starts CRCReceiveTimer .	Physical Layer receives the Wait Message and compares the CRC it calculated with the one sent to verify the Message.
12		Physical Layer forwards the Wait Message to the Protocol Layer.

13		<p>Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value.</p> <p>Protocol Layer informs the Policy Engine that an <i>Wait</i> Message has been received. The Policy Engine informs the Device Policy Manager that the Request has been rejected.</p>
14		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
15	Physical Layer receives the <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
16	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops the <i>CRCReceiveTimer</i> .	
17	The Protocol Layer informs the Policy Engine that a <i>Wait</i> Message was successfully sent.	

8.3.2.2.2 EPR

8.3.2.2.2.1 Entering EPR Mode

8.3.2.2.2.1.1 Entering EPR Mode (Success)

This is an example of an Enter EPR Mode operation where the Sink requests EPR mode when this process succeeds. [Figure 8-13 “Entering EPR Mode \(Success\)”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Enter EPR process.

 [Figure 8-13 “Entering EPR Mode \(Success\)”](#)

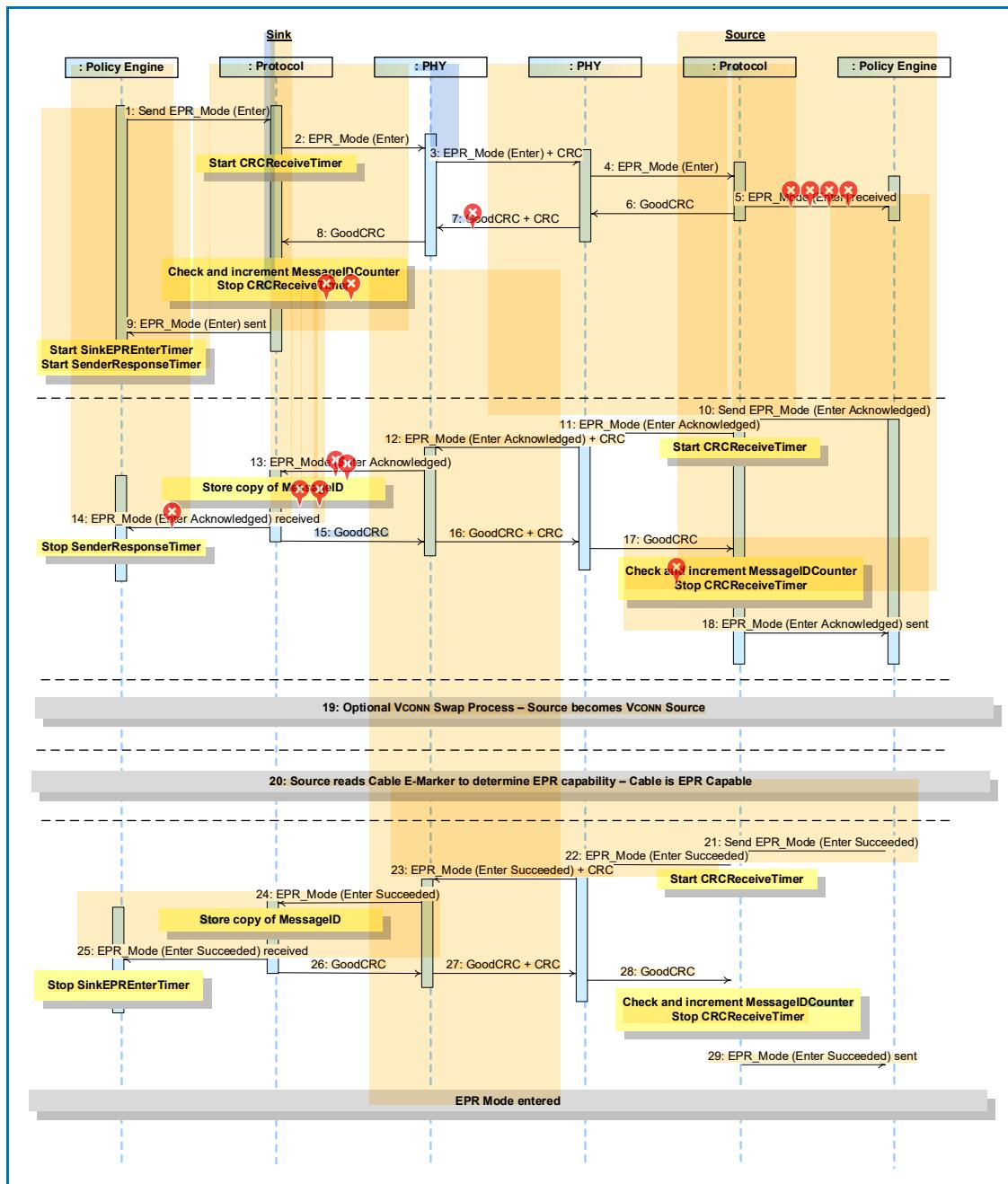


Table 8.41 “Steps for Entering EPR Mode (Success)” below provides a detailed explanation of what happens at each labeled step in **Figure 8-13 “Entering EPR Mode (Success)”** above.

Table 8.41 “Steps for Entering EPR Mode (Success)”

Step	Sink	Source
1	The Policy Engine directs the Protocol Layer to generate an EPR_Mode (Enter) Message to request entry to EPR mode.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the EPR_Mode (Enter) Message. Starts CRCReceiveTimer .	Physical Layer receives the EPR_Mode (Enter) Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the EPR_Mode (Enter) Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Mode (Enter) Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the EPR_Mode (Enter) Source_Capabilities Message was successfully sent. The Policy Engine starts the SenderResponseTimer and the SinkEPREnEnterTimer .	
10		Policy Engine evaluates the EPR_Mode (Enter) Message sent by the Sink. It tells the Protocol Layer to form a EPR_Mode (Enter Acknowledged) Message.
11		Protocol Layer creates the EPR_Mode (Enter Acknowledged) Message and passes to Physical Layer.
12	Physical Layer receives the EPR_Mode (Enter Acknowledged) Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the EPR_Mode (Enter Acknowledged) Message. Starts CRCReceiveTimer .
13	Physical Layer removes the CRC and forwards the EPR_Mode (Enter Acknowledged) Message to the Protocol Layer.	
14	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the EPR_Mode (Enter Acknowledged) information to the Policy Engine. The Policy Engine stops the SenderResponseTimer .	

15	The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.	
16	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		The protocol Layer verifies and increments the <i>MessageIDCounter</i> . It informs the Policy Engine that the <i>EPR_Mode</i> (Enter Acknowledged) Message was successfully sent. The Protocol Layer stops the <i>CRCReceiveTimer</i> .
19	If the Source is not the VCONN Source the Source initiates the VCONN swap process as described in Section 8.3.2.11 "Vconn Swap".	
20	The Source performs cable discovery to determine whether the cable supports EPR. The Cable Discovery process is described in Section 8.3.2.15.1 "Discover Identity".	
21		The Source is now the VCONN Source and has determined that the Sink and the cable are EPR capable. The Policy Engine tells the Protocol Layer to form a <i>EPR_Mode</i> (Enter Succeeded) Message.
22		Protocol Layer creates the <i>EPR_Mode</i> (Enter Succeeded) Message and passes to Physical Layer.
23	Physical Layer receives the <i>EPR_Mode</i> (Enter Succeeded) Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>EPR_Mode</i> (Enter Succeeded) Message. Starts <i>CRCReceiveTimer</i> .
24	Physical Layer removes the CRC and forwards the <i>EPR_Mode</i> (Enter Succeeded) Message to the Protocol Layer.	
25	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the <i>EPR_Mode</i> (Enter Succeeded) information to the Policy Engine. The Policy Engine stops the <i>SinkEPREnEnterTimer</i> .	
26	The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.	
27	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
28		Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer.
29		The protocol Layer verifies and increments the <i>MessageIDCounter</i> . It informs the Policy Engine that the <i>EPR_Mode</i> (Enter Succeeded) Message was successfully sent. The Protocol Layer stops the <i>CRCReceiveTimer</i> .
EPR Mode Entered		

8.3.2.2.2.1.2

Entering EPR Mode (Failure due to non-EPR cable)

This is an example of an Enter EPR Mode operation where the Sink requests EPR mode when this process fails due to the cable not being capable of EPR. [Figure 8-14 “Entering EPR Mode \(Failure due to non-EPR cable\)»](#) shows the Messages as they flow across the bus and within the devices to accomplish the Enter EPR process.

[Figure 8-14 “Entering EPR Mode \(Failure due to non-EPR cable\)»](#)

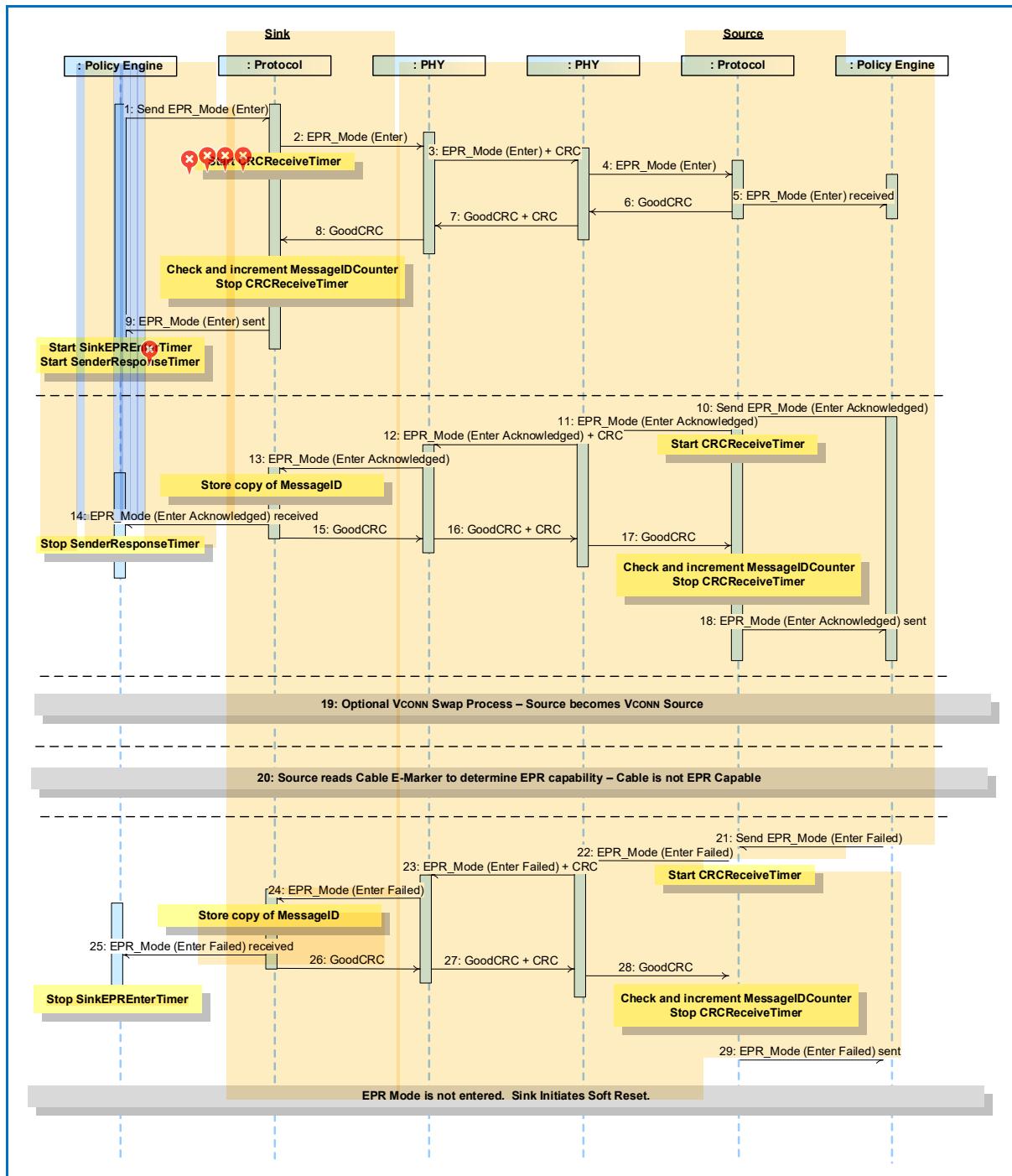


Table 8.42 “Steps for Entering EPR Mode (Failure due to non-EPR cable)” below provides a detailed explanation of what happens at each labeled step in **Figure 8-14 “Entering EPR Mode (Failure due to non-EPR cable)”** above.

Table 8.42 “Steps for Entering EPR Mode (Failure due to non-EPR cable)”

Step	Sink	Source
1	The Policy Engine directs the Protocol Layer to generate an EPR_Mode (Enter) Message to request entry to EPR mode.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the EPR_Mode (Enter) Message. Starts CRCReceiveTimer .	Physical Layer receives the EPR_Mode (Enter) Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the EPR_Mode (Enter) Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Mode (Enter) Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the EPR_Mode (Enter) Message was successfully sent. The Policy Engine starts the SenderResponseTimer and the SinkEPREEnterTimer .
10		Policy Engine evaluates the EPR_Mode (Enter) Message sent by the Sink. It tells the Protocol Layer to form a EPR_Mode (Enter Acknowledged) Message.
11		Protocol Layer creates the EPR_Mode (Enter Acknowledged) Message and passes to Physical Layer.
12	Physical Layer receives the EPR_Mode (Enter Acknowledged) Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the EPR_Mode (Enter Acknowledged) Message. Starts CRCReceiveTimer .
13	Physical Layer removes the CRC and forwards the EPR_Mode (Enter Acknowledged) Message to the Protocol Layer.	
14	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the EPR_Mode (Enter Acknowledged) information to the Policy Engine. The Policy Engine stops the SenderResponseTimer .	

15	The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.	
16	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		The protocol Layer verifies and increments the <i>MessageIDCounter</i> . It informs the Policy Engine that the <i>EPR_Mode</i> (Enter Acknowledged) Message was successfully sent. The Protocol Layer stops the <i>CRCReceiveTimer</i> .
19	If the Source is not the VCONN Source the Source initiates the VCONN swap process as described in Section 8.3.2.11 "Vconn Swap" .	
20	The Source performs cable discovery to determine whether the cable supports EPR; cable is not EPR capable. The Cable Discovery process is described in Section 8.3.2.15.1 "Discover Identity" .	
21		The Source determines that there has been a failure or incompatibility during the EPR process (see Section 6.4.10 "EPR_Mode Message"). The Policy Engine tells the Protocol Layer to form a <i>EPR_Mode</i> (Enter Failed) Message.
22		Protocol Layer creates the <i>EPR_Mode</i> (Enter Failed) Message and passes to Physical Layer.
23	Physical Layer receives the <i>EPR_Mode</i> (Enter Failed) Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>EPR_Mode</i> (Enter Failed) Message. Starts <i>CRCReceiveTimer</i> .
24	Physical Layer removes the CRC and forwards the <i>EPR_Mode</i> (Enter Failed) Message to the Protocol Layer.	
25	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the <i>EPR_Mode</i> (Enter Failed) information to the Policy Engine. The Policy Engine stops the <i>SinkEPREnterTimer</i> .	
26	The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.	
27	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
28		Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer.
29		The protocol Layer verifies and increments the <i>MessageIDCounter</i> . It informs the Policy Engine that the <i>EPR_Mode</i> (Enter Failed) Message was successfully sent. The Protocol Layer stops the <i>CRCReceiveTimer</i> .
EPR Mode is not entered. Sink Initiates Soft Reset		

8.3.2.2.2.1.3

Entering EPR Mode (Failure of VCONN Swap)

This is an example of an Enter EPR Mode operation where the Sink requests EPR mode when this process fails due to a failure of the VCONN Swap process. [Figure 8-15 “Entering EPR Mode \(Failure of Vconn Swap\)”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Enter EPR process.

[Figure 8-15 “Entering EPR Mode \(Failure of Vconn Swap\)”](#)

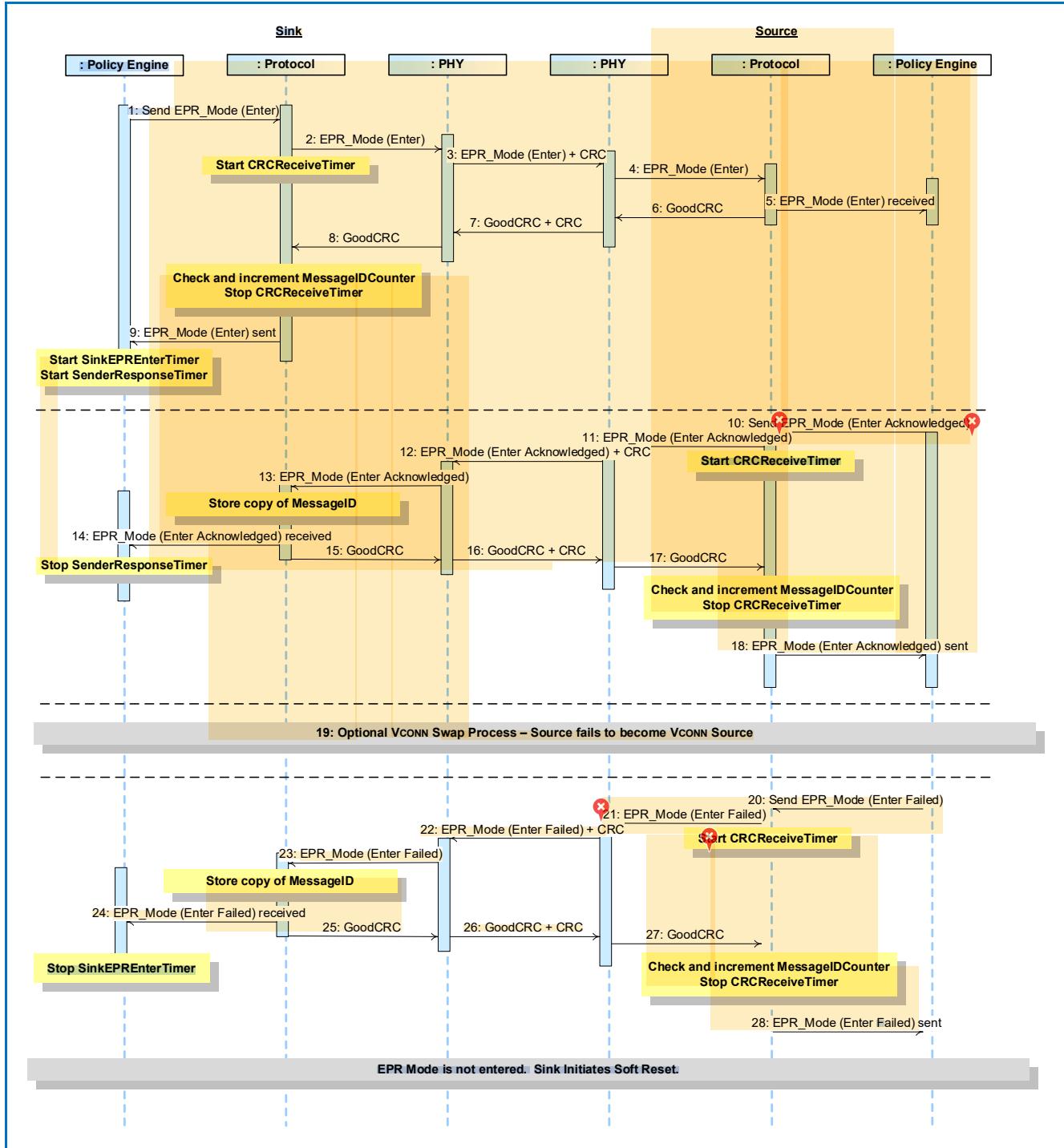


Table 8.43 "Steps for Entering EPR Mode (Failure of Vconn Swap)" below provides a detailed explanation of what happens at each labeled step in **Figure 8-15 "Entering EPR Mode (Failure of VCONN Swap)"** above.

Table 8.43 "Steps for Entering EPR Mode (Failure of VCONN Swap)"

Step	Sink	Source
1	The Policy Engine directs the Protocol Layer to generate an EPR_Mode (Enter) Message to request entry to EPR mode.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the EPR_Mode (Enter) Message. Starts CRCReceiveTimer .	Physical Layer receives the EPR_Mode (Enter) Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the EPR_Mode (Enter) Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Mode (Enter) Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the EPR_Mode (Enter) Message was successfully sent. The Policy Engine starts the SenderResponseTimer and the SinkEPREEnterTimer .
10		Policy Engine evaluates the EPR_Mode (Enter) Message sent by the Sink. It tells the Protocol Layer to form a EPR_Mode (Enter Acknowledged) Message.
11		Protocol Layer creates the EPR_Mode (Enter Acknowledged) Message and passes to Physical Layer.
12	Physical Layer receives the EPR_Mode (Enter Acknowledged) Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the EPR_Mode (Enter Acknowledged) Message. Starts CRCReceiveTimer .
13	Physical Layer removes the CRC and forwards the EPR_Mode (Enter Acknowledged) Message to the Protocol Layer.	
14	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the EPR_Mode (Enter Acknowledged) information to the Policy Engine. The Policy Engine stops the SenderResponseTimer .	

15	The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.	
16	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer forwards the GoodCRC Message to the Protocol Layer.
18		The protocol Layer verifies and increments the MessageIDCounter . It informs the Policy Engine that the EPR_Mode (Enter Acknowledged) Message was successfully sent. The Protocol Layer stops the CRCReceiveTimer .
19	✖ If the Source is not the VCONN Source the Source initiates the VCONN swap process as described in Section 8.3.2.11 "Vconn Swap" . In this case the VCONN swap process fails.	
20		The Source determines that there has been a failure or incompatibility during the EPR process (see Section 6.4.10 "EPR_Mode Message"). The Policy Engine tells the Protocol Layer to form a EPR_Mode (Enter Failed) Message.
21		Protocol Layer creates the EPR_Mode (Enter Failed) Message and passes to Physical Layer.
22	Physical Layer receives the EPR_Mode (Enter Failed) Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the EPR_Mode (Enter Failed) Message. Starts CRCReceiveTimer .
23	Physical Layer removes the CRC and forwards the EPR_Mode (Enter Failed) Message to the Protocol Layer.	
24	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the EPR_Mode (Enter Failed) information to the Policy Engine. The Policy Engine stops the SinkEPREnterTimer .	
25	The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.	
26	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
27		Physical Layer forwards the GoodCRC Message to the Protocol Layer.
28		The protocol Layer verifies and increments the MessageIDCounter . It informs the Policy Engine that the EPR_Mode (Enter Failed) Message was successfully sent. The Protocol Layer stops the CRCReceiveTimer .
EPR Mode is not entered. Sink Initiates Soft Reset		

8.3.2.2.2.2

EPR Explicit Contract Negotiation

8.3.2.2.2.2.1

EPR Explicit Contract Negotiation (Accept)

Figure 8-16 “Successful Fixed EPR Power Negotiation” illustrates an example of a successful Message flow while negotiating an Explicit Contract in EPR Mode. The negotiation goes through several distinct phases:

- The Source sends out its power capabilities in an ***EPR_Source_Capabilities*** Message.
- The Sink evaluates these capabilities and, in the request phase, selects one power level by sending an ***EPR_Request*** Message.
- The Source evaluates the request and accepts the request with an ***Accept*** Message.
- The Source transitions to the new power level and then informs the Sink by sending a ***PS_RDY*** Message.
- The Sink starts using the new power level.
- the Source starts its keep alive timer
- the Sink starts its request timer to send periodic ***EPR_KeepAlive*** Messages

Figure 8-16 "Successful Fixed EPR Power Negotiation"

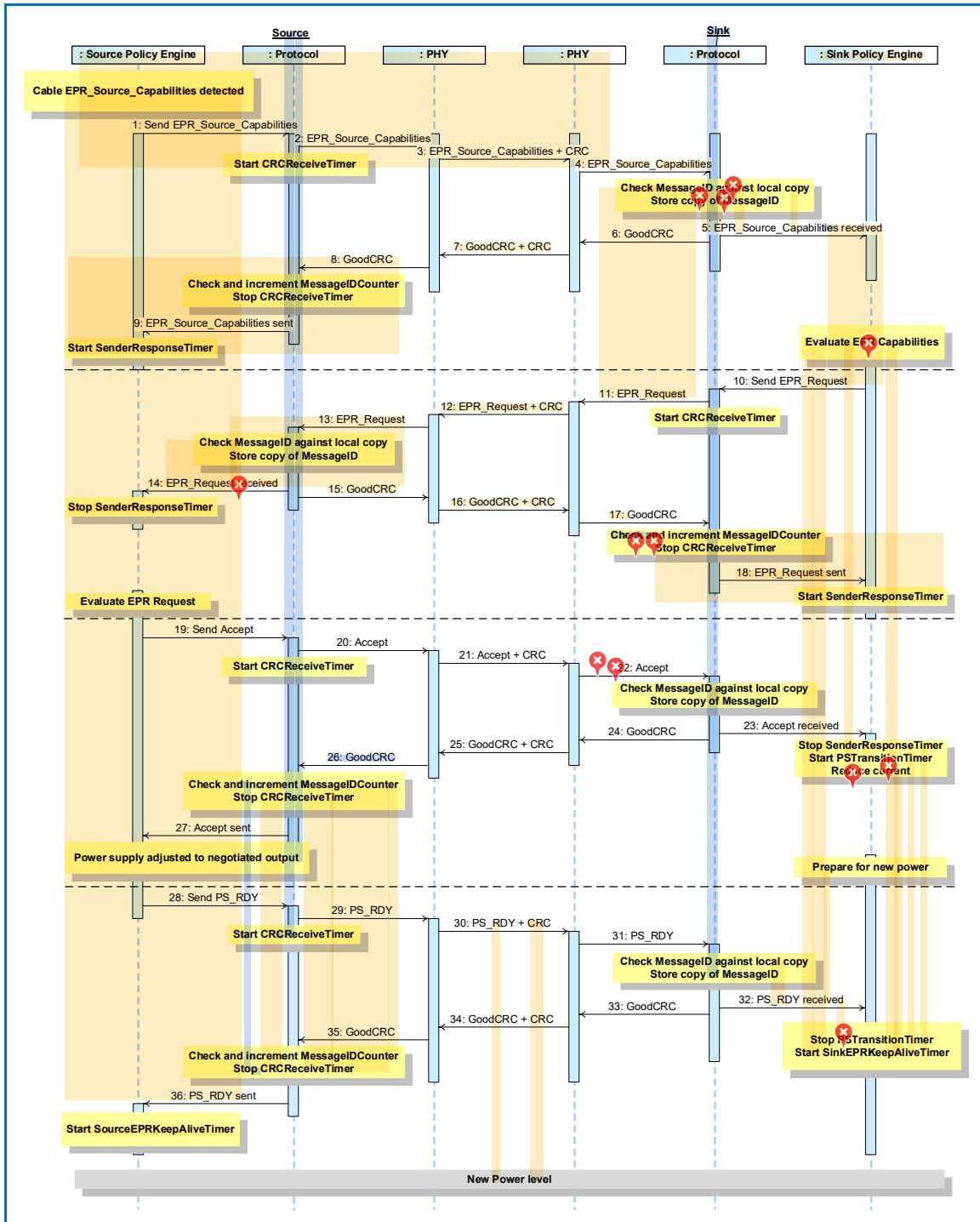


Table 8.44 "Steps for a successful EPR Power Negotiation" below provides a detailed explanation of what happens at each labeled step in **Figure 8-16 "Successful Fixed EPR Power Negotiation"** above.

Table 8.44 "Steps for a successful EPR Power Negotiation"

Step	Source	Sink
1	The Cable Capabilities are detected if these are not already known (see Section 4.4 "Cable Type Detection"). Policy Engine directs the Protocol Layer to send a EPR_Source_Capabilities Message that represents the power supply's present capabilities.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the EPR_Source_Capabilities Message. Starts CRCReceiveTimer .	Physical Layer receives the EPR_Source_Capabilities Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the EPR_Source_Capabilities Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Source_Capabilities Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the EPR_Source_Capabilities Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the EPR_Source_Capabilities Message sent by the Source and selects which power it would like. It tells the Protocol Layer to form the data (e.g., Power Data Object) that represents its Request into a Message.
11		Protocol Layer creates the EPR_Request Message and passes to Physical Layer.
12	Physical Layer receives the EPR_Request Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the EPR_Request Message. Starts CRCReceiveTimer .
13	Physical Layer removes the CRC and forwards the EPR_Request Message to the Protocol Layer.	

14	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine. Policy Engine stops <i>SenderResponseTimer</i> .	
15	The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.	
16	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		The protocol Layer verifies and increments the <i>MessageIDCounter</i> . It informs the Policy Engine that the <i>EPR_Request</i> Message was successfully sent. The Protocol Layer stops the <i>CRCReceiveTimer</i> . The Policy Engine starts <i>SenderResponseTimer</i> .
19	Policy Engine evaluates the <i>EPR_Request</i> Message sent by the Sink and decides if it can meet the request. It tells the Protocol Layer to form an <i>Accept</i> Message.	
20	The Protocol Layer forms the <i>Accept</i> Message that is passed to the Physical Layer.	
21	Physical Layer appends CRC and sends the <i>Accept</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer forwards the <i>Accept</i> Message to the Protocol Layer.
23		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that an <i>Accept</i> Message has been received. The Policy Engine stops <i>SenderResponseTimer</i> , starts the <i>PSTransitionTimer</i> and reduces its current draw. The Device Policy Manager prepares the Power supply for transition to the new power level.
24		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
25	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Message.
26	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops the <i>CRCReceiveTimer</i> .	
27	The Protocol Layer informs the Policy Engine that an <i>Accept</i> Message was successfully sent.	
Power supply Adjusts its Output to the Negotiated Value		

28	The Device Policy Manager informs the Policy Engine that the power supply has settled at the new operating condition and tells the Protocol Layer to send a <i>PS_RDY</i> Message.	
29	The Protocol Layer forms the <i>PS_RDY</i> Message.	
30	Physical Layer appends CRC and sends the <i>PS_RDY</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the <i>PS_RDY</i> Message and compares the CRC it calculated with the one sent to verify the Message.
31		Physical Layer forwards the <i>PS_RDY</i> Message to the Protocol Layer.
32		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a RS_RDY has been received. The Policy Engine stops the <i>PSTransitionTimer</i> . The Policy Engine starts the <i>SinkEPRKeepAliveTimer</i> .
33		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
34	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message. *	Physical Layer appends CRC and sends the Message.
35	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> . Stops the <i>CRCReceiveTimer</i> .	
36	The Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent.	
37	When in EPR operation the Policy Engine starts the <i>SourceEPRKeepAliveTimer</i> .	

8.3.2.2.2.2.2

EPR Explicit Contract Negotiation (Reject)

Figure 8-17 “Rejected Fixed EPR Power Negotiation” illustrates an example of a Message flow where the request is rejected while negotiating an Explicit Contract in EPR Mode. The negotiation goes through several distinct phases:

- The Source sends out its power capabilities in an **EPR_Source_Capabilities** Message.
- The Sink evaluates these capabilities and, in the request phase, selects one power level by sending an **EPR_Request** Message.
- The Source evaluates the request and accepts the request with a **Reject** Message.

Figure 8-17 “Rejected Fixed EPR Power Negotiation”

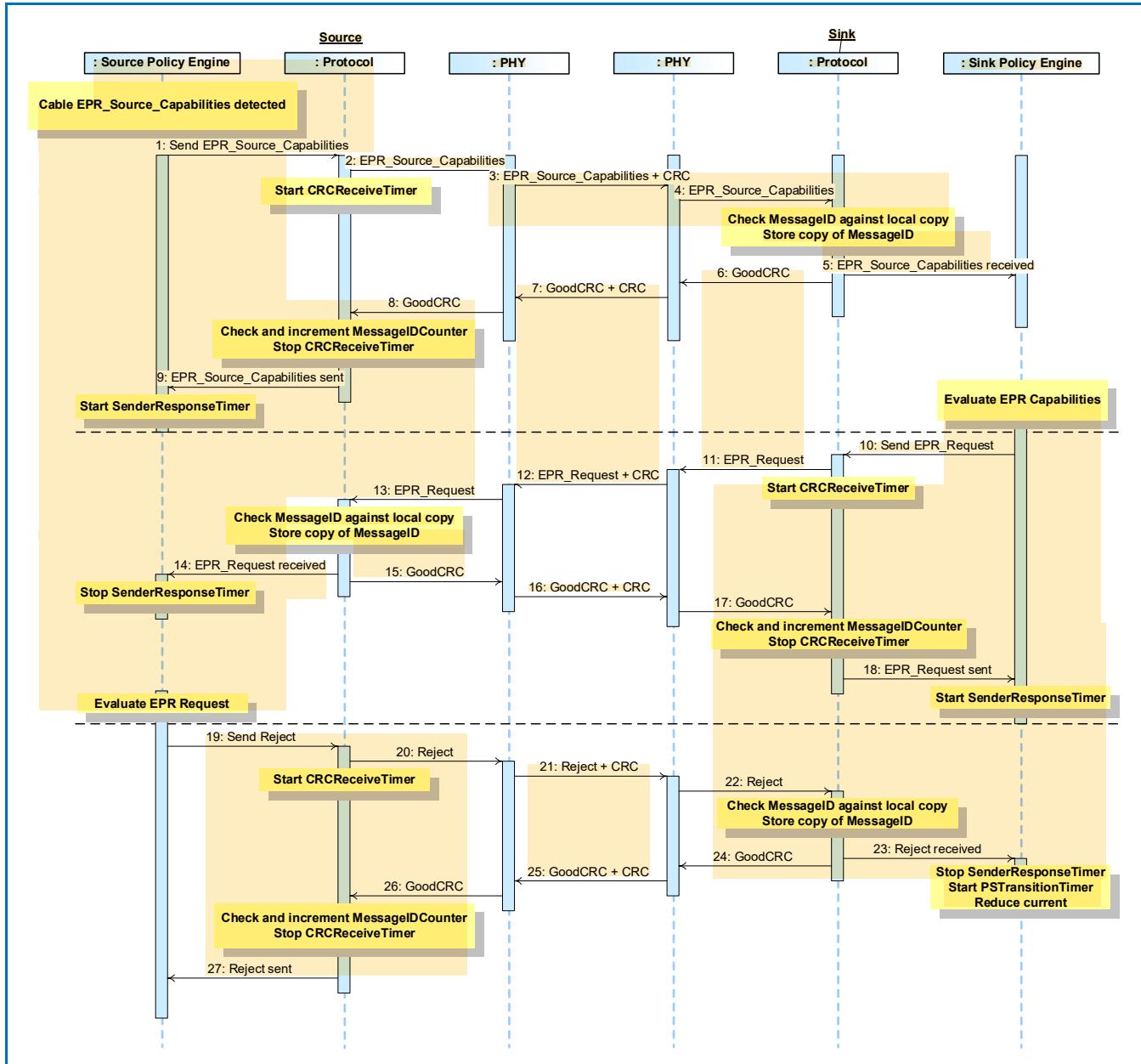


Table 8.45 “Steps for a Rejected EPR Power Negotiation” below provides a detailed explanation of what happens at each labeled step in **Figure 8-17 “Rejected Fixed EPR Power Negotiation”** above.

Table 8.45 “Steps for a Rejected EPR Power Negotiation”

Step	Source	Sink
1	The Cable Capabilities are detected if these are not already known (see Section 4.4 “Cable Type Detection”). Policy Engine directs the Protocol Layer to send a EPR_Source_Capabilities Message that represents the power supply's present capabilities.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Source_Capabilities Message. Starts CRCReceiveTimer .	Physical Layer receives the EPR_Source_Capabilities Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the EPR_Source_Capabilities Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Source_Capabilities Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the EPR_Source_Capabilities Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the EPR_Source_Capabilities Message sent by the Source and selects which power it would like. It tells the Protocol Layer to form the data (e.g., Power Data Object) that represents its Request into a Message.
11		Protocol Layer creates the EPR_Request Message and passes to Physical Layer.
12	Physical Layer receives the EPR_Request Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the EPR_Request Message. Starts CRCReceiveTimer .
13	Physical Layer removes the CRC and forwards the EPR_Request Message to the Protocol Layer.	

14	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine. Policy Engine stops SenderResponseTimer .	
15	The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.	
16	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer forwards the GoodCRC Message to the Protocol Layer.
18		The protocol Layer verifies and increments the MessageIDCounter . It informs the Policy Engine that the EPR_Request Message was successfully sent. The Protocol Layer stops the CRCReceiveTimer . The Policy Engine starts SenderResponseTimer .
19	Policy Engine evaluates the EPR_Request Message sent by the Sink and decides if it can meet the request. It tells the Protocol Layer to form a Reject Message.	
20	The Protocol Layer forms the Reject Message that is passed to the Physical Layer.	
21	Physical Layer appends CRC and sends the Reject Message. Starts CRCReceiveTimer .	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer forwards the Reject Message to the Protocol Layer.
23		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a Reject Message has been received. The Policy Engine stops SenderResponseTimer .
24		The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.
25	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Message.
26	Physical Layer forwards the GoodCRC Message to the Protocol Layer. The Protocol Layer verifies and increments the MessageIDCounter and stops the CRCReceiveTimer .	
27	The Protocol Layer informs the Policy Engine that a Reject Message was successfully sent.	

8.3.2.2.2.2.3

EPR Explicit Contract Negotiation (Wait)

Figure 8-18 “Wait response to Fixed EPR Power Negotiation” illustrates an example of a Message flow where the request is responded to with wait while negotiating an Explicit Contract in EPR Mode. The negotiation goes through several distinct phases:

- The Source sends out its power capabilities in an **EPR_Source_Capabilities** Message.
- The Sink evaluates these capabilities and, in the request phase, selects one power level by sending an **EPR_Request** Message.
- The Source evaluates the request and accepts the request with a **Wait** Message.

Figure 8-18 “Wait response to Fixed EPR Power Negotiation”

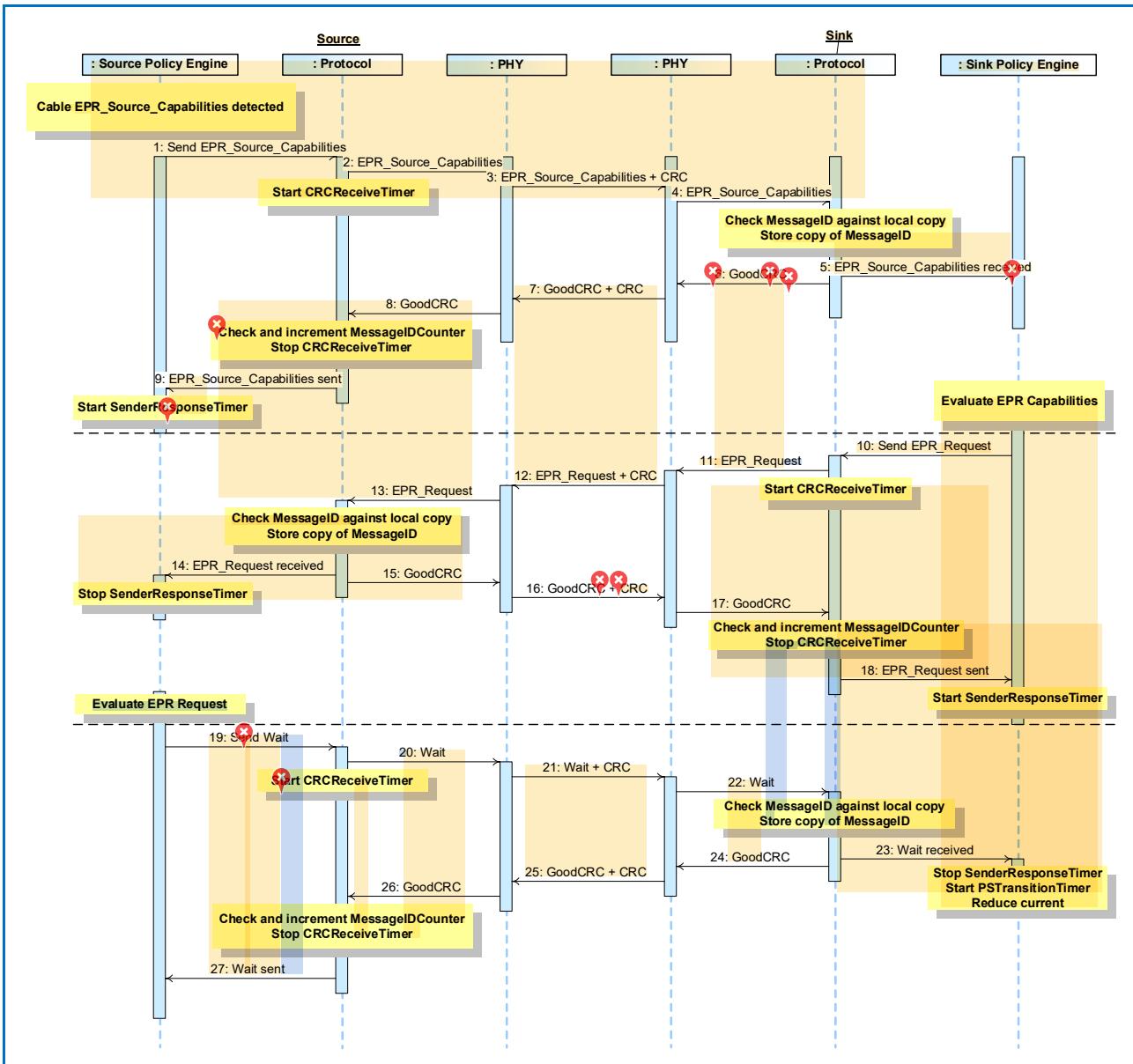


Table 8.46 "Steps for a Wait response to an EPR Power Negotiation" below provides a detailed explanation of what happens at each labeled step in Figure 8-18 "Wait response to Fixed EPR Power Negotiation" above.

Table 8.46 "Steps for a Wait response to an EPR Power Negotiation"

Step	Source	Sink
1	The Cable Capabilities are detected if these are not already known (see Section 4.4 "Cable Type Detection"). Policy Engine directs the Protocol Layer to send a EPR_Source_Capabilities Message that represents the power supply's present capabilities.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Source_Capabilities Message. Starts CRCReceiveTimer .	Physical Layer receives the EPR_Source_Capabilities Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the EPR_Source_Capabilities Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Source_Capabilities Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the EPR_Source_Capabilities Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the EPR_Source_Capabilities Message sent by the Source and selects which power it would like. It tells the Protocol Layer to form the data (e.g., Power Data Object) that represents its Request into a Message.
11		Protocol Layer creates the EPR_Request Message and passes to Physical Layer.
12	Physical Layer receives the EPR_Request Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the EPR_Request Message. Starts CRCReceiveTimer .
13	Physical Layer removes the CRC and forwards the EPR_Request Message to the Protocol Layer.	

14	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine. Policy Engine stops <i>SenderResponseTimer</i> .	
15	The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.	
16	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		The protocol Layer verifies and increments the <i>MessageIDCounter</i> . It informs the Policy Engine that the <i>EPR_Request</i> Message was successfully sent. The Protocol Layer stops the <i>CRCReceiveTimer</i> . The Policy Engine starts <i>SenderResponseTimer</i> .
19	Policy Engine evaluates the <i>EPR_Request</i> Message sent by the Sink and decides if it can meet the request. It tells the Protocol Layer to form a <i>Wait</i> Message.	
20	The Protocol Layer forms the <i>Wait</i> Message that is passed to the Physical Layer.	
21	Physical Layer appends CRC and sends the <i>Wait</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer forwards the <i>Wait</i> Message to the Protocol Layer.
23		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a <i>Wait</i> Message has been received. The Policy Engine stops <i>SenderResponseTimer</i> .
24		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
25	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Message.
26	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops the <i>CRCReceiveTimer</i> .	
27	The Protocol Layer informs the Policy Engine that a <i>Wait</i> Message was successfully sent.	

8.3.2.2.2.3

EPR Keep Alive

This is an example of keep alive operation during an Explicit Contract in EPR Mode. [Figure 8-19 “EPR Keep Alive”](#) shows the Messages as they flow across the bus and within the devices to accomplish the keep alive.

[Figure 8-19 “EPR Keep Alive”](#)

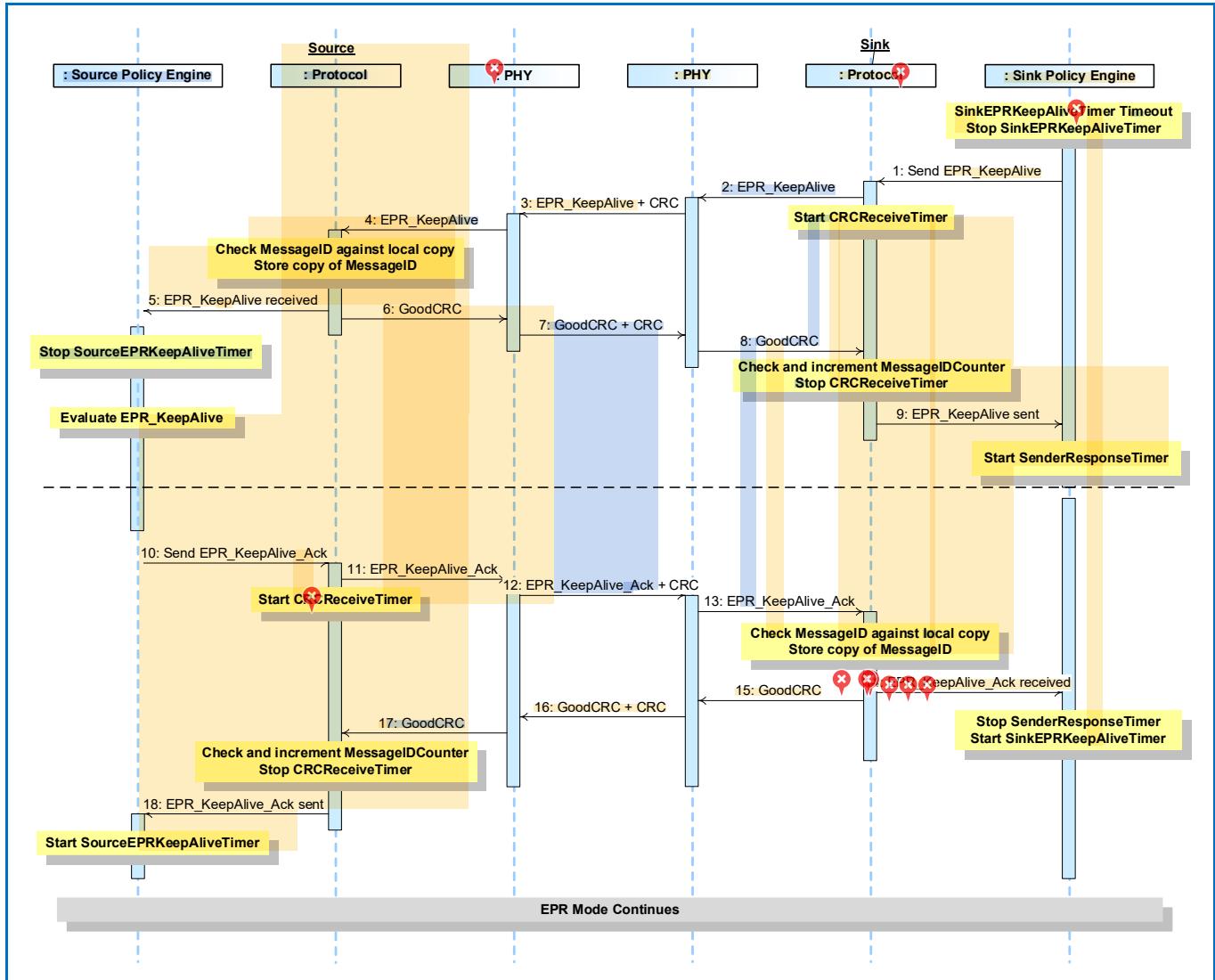




Table 8.47 “Steps for EPR Keep Alive” below provides a detailed explanation of what happens at each labeled step in **Figure 8-19 “EPR Keep Alive”** above.

Table 8.47 “Steps for EPR Keep Alive”

Step	Source	Sink
1		The SinkEPRKeepAliveTimer times out in the Policy Engine. The Policy Engine stops the SinkEPRKeepAliveTimer timer and tells the Protocol Layer to form an EPR_KeepAlive Message.
2		The Protocol Layer creates the EPR_KeepAlive Message and passes it to Physical Layer. The Protocol Layer.
3	Physical Layer receives the EPR_KeepAlive Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Request Message. Starts CRCReceiveTimer .
4	Physical Layer removes the CRC and forwards the EPR_KeepAlive Message to the Protocol Layer.	
5	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine. Policy Engine stops the SourceEPRKeepAliveTimer .	
6	The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.	
7	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
8		Physical Layer forwards the GoodCRC Message to the Protocol Layer.
9		The protocol Layer verifies and increments the MessageIDCounter . It informs the Policy Engine that the SinkEPRKeepAliveTimer Message was successfully sent. The Protocol Layer stops the CRCReceiveTimer . The Policy Engine starts SenderResponseTimer .
10	Policy Engine requests the Device Policy Manager to evaluate the SourceEPRKeepAliveTimer Message sent by the Sink and decides if the Source can meet the request. The Policy Engine tells the Protocol Layer to form an EPR_KeepAlive_Ack Message.	
11	The Protocol Layer forms the EPR_KeepAlive_Ack Message that is passed to the Physical Layer.	
12	Physical Layer appends CRC and sends the EPR_KeepAlive_Ack Message. Starts CRCReceiveTimer .	Physical Layer receives the EPR_KeepAlive_Ack Message and compares the CRC it calculated with the one sent to verify the Message.
13		Physical Layer forwards the EPR_KeepAlive_Ack Message to the Protocol Layer.

14		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that an <i>Accept</i> Message has been received. The Policy Engine stops <i>SenderResponseTimer</i> , starts the <i>SinkEPRKeepAliveTimer</i> .
15		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
16	Physical Layer receives the <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
17	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops the <i>CRCReceiveTimer</i> .	
18	The Protocol Layer informs the Policy Engine that an <i>EPR_KeepAlive_Ack</i> Message was successfully sent. The Policy Engine starts the SourceEPRKeepAliveTimer.	

✖ EPR Mode Continues

8.3.2.2.2.4

Exiting EPR Mode

8.3.2.2.2.4.1

Exiting EPR Mode (Sink Initiated)

This is an example of an Exit EPR Mode operation where the Sink requests EPR mode to be exited. [Figure 8-20 “Exiting EPR Mode \(Sink Initiated\)”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Exit EPR process.

Figure 8-20 “Exiting EPR Mode (Sink Initiated)”

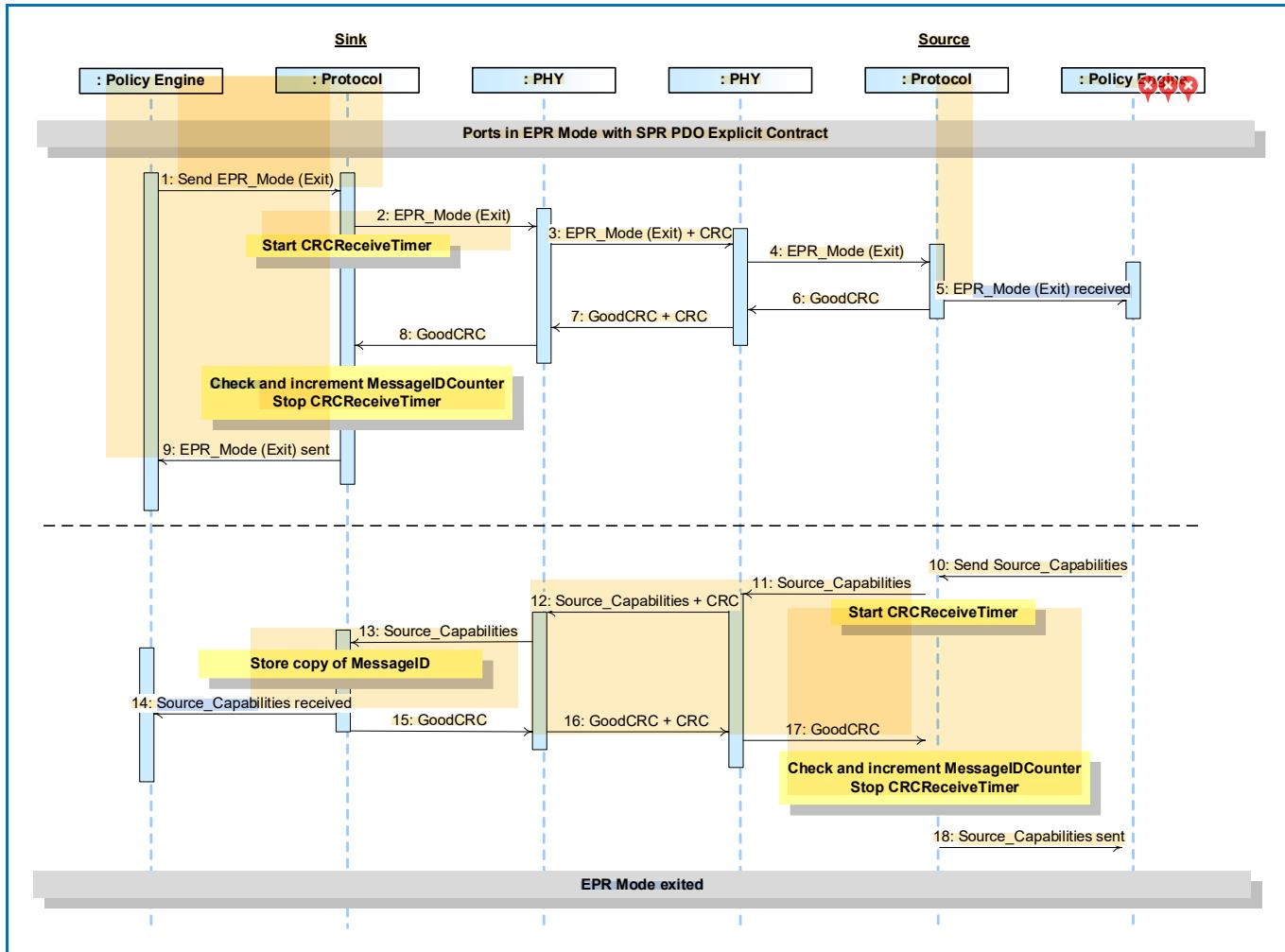


Table 8.48 "Steps for Exiting EPR Mode (Sink Initiated)" below provides a detailed explanation of what happens at each labeled step in **Figure 8-20 "Exiting EPR Mode (Sink Initiated)"** above.

Table 8.48 "Steps for Exiting EPR Mode (Sink Initiated)"

Step	Sink	Source
The Port Partners are in an Explicit Contract using an SPR (A)PDO (Voltage <= 20V)		
1	The Policy Engine directs the Protocol Layer to generate an EPR_Mode (Exit) Message to request entry to EPR mode.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the EPR_Mode (Exit) Message. Starts CRCReceiveTimer .	Physical Layer receives the EPR_Mode (Exit) Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the EPR_Mode (Exit) Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Mode (Exit) Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it to Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the EPR_Mode (Exit) Message was successfully sent.	
10		Policy Engine evaluates the EPR_Mode (Exit) Message sent by the Sink. It tells the Protocol Layer to form a Source_Capabilities Message.
11		Protocol Layer creates the Source_Capabilities Message and passes to Physical Layer.
12	Physical Layer receives the Source_Capabilities Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Source_Capabilities Message. Starts CRCReceiveTimer .
13	Physical Layer removes the CRC and forwards the Source_Capabilities Message to the Protocol Layer.	
14	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the EPR_Mode (Enter Succeeded) information to the Policy Engine.	
15	The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.	

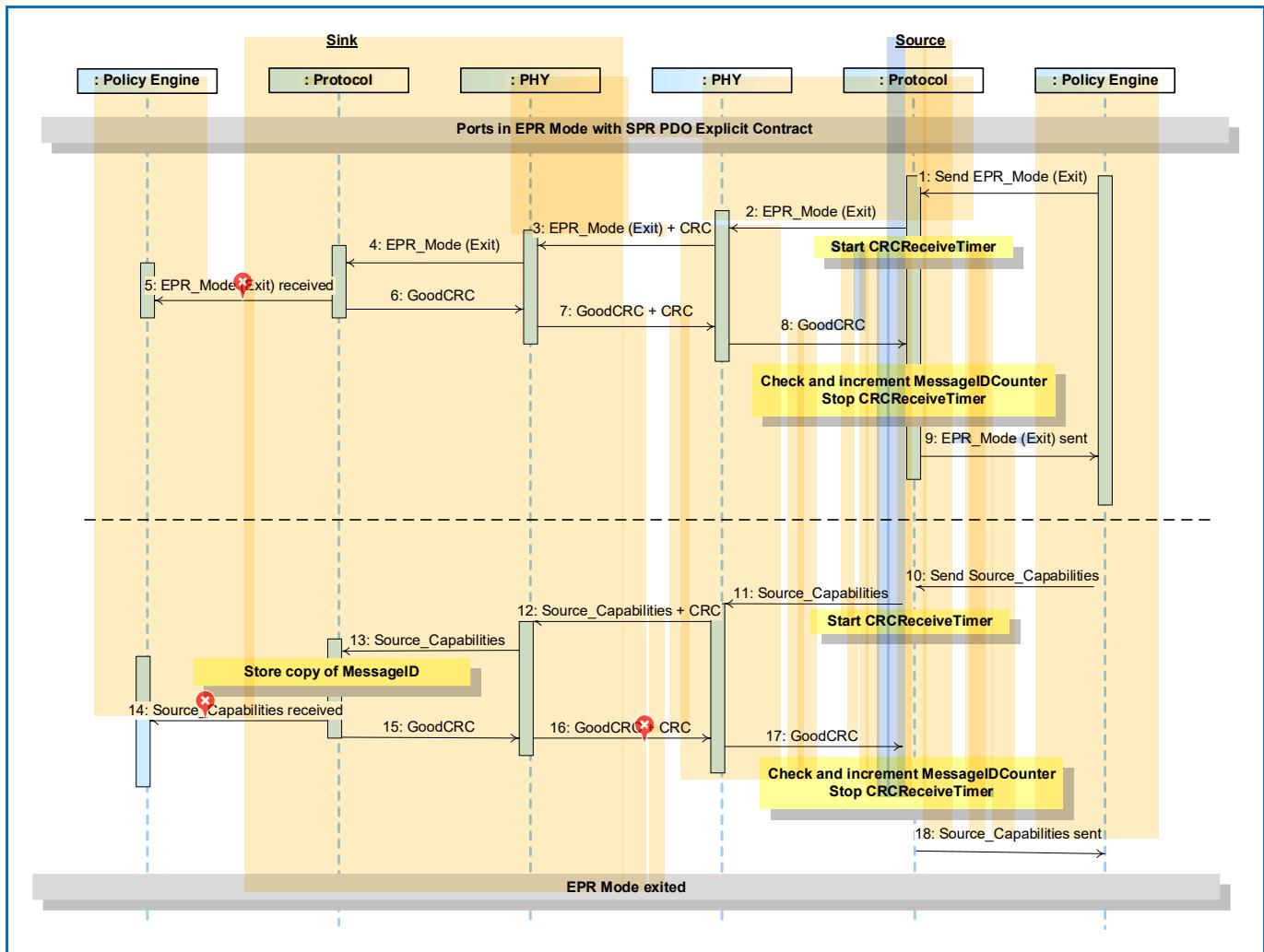
16	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer forwards the GoodCRC Message to the Protocol Layer.
18		The protocol Layer verifies and increments the MessageIDCounter . It informs the Policy Engine that the Source_Capabilities Message was successfully sent. The Protocol Layer stops the CRCReceiveTimer .
EPR Mode Exited. Power Negotiation proceeds as defined in Section 8.3.2.2.1.1 "SPR Explicit Contract Negotiation" .		

8.3.2.2.2.4.2

Exiting EPR Mode (Source Initiated)

This is an example of an Exit EPR Mode operation where the Source requests EPR mode to be exited. [Figure 8-21 “Exiting EPR Mode \(Source Initiated\)”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Exit EPR process.

 [Figure 8-21 “Exiting EPR Mode \(Source Initiated\)”](#)



✖**Table 8.49 “Steps for Exiting EPR Mode (Source Initiated)”** below provides a detailed explanation of what happens at each labeled step in **Figure 8-21 “Exiting EPR Mode (Source Initiated)”** above.

Table 8.49 “Steps for Exiting EPR Mode (Source Initiated)”

Step	Sink	Source
The Port Partners are in an Explicit Contract using an SPR (A)PDO (Voltage <= 20V)		
1		The Policy Engine directs the Protocol Layer to generate an EPR_Mode (Exit) Message to request entry to EPR mode.
2		Protocol Layer creates the Message and passes to Physical Layer.
3	Physical Layer receives the EPR_Mode (Exit) Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the EPR_Mode (Exit) Message. Starts CRCReceiveTimer .
4	Physical Layer removes the CRC and forwards the EPR_Mode (Exit) Message to the Protocol Layer.	
✖5	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Mode (Exit) Message information to the Policy Engine that consumes it.	
6	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
7	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.
8		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
9		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the EPR_Mode (Exit) Message was successfully sent.
10		Policy Engine evaluates the EPR_Mode (Exit) Message sent by the Sink. It tells the Protocol Layer to form a Source_Capabilities Message.
11		Protocol Layer creates the Source_Capabilities Message and passes to Physical Layer. Starts CRCReceiveTimer .
12	Physical Layer receives the Source_Capabilities Message and compares the CRC it calculated with the one sent to verify the Message.✖	Physical Layer appends a CRC and sends the Source_Capabilities Message.
13	Physical Layer removes the CRC and forwards the Source_Capabilities Message to the Protocol Layer.	
14	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the EPR_Mode (Enter Succeeded) information to the Policy Engine.	

15	The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.	
16	Physical Layer appends CRC and sends the Message.	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		The protocol Layer verifies and increments the <i>MessageIDCounter</i> . It informs the Policy Engine that the <i>Source_Capabilities</i> Message was successfully sent. The Protocol Layer stops the <i>CRCReceiveTimer</i> .
EPR Mode Exited. Power Negotiation proceeds as defined in <i>Section 8.3.2.2.1.1 “SPR Explicit Contract Negotiation”</i> .		



8.3.2.2.2.5

EPR Sink Makes Request

8.3.2.2.2.5.1

EPR Sink Makes Request (Accept)

This is an example of EPR when a Sink makes a Request which is Accepted during an Explicit Contract. [Figure 8-22 “EPR Sink Makes Request \(Accept\)”](#) shows the Messages as they flow across the bus and within the devices to accomplish the keep alive. *

Figure 8-22 “EPR Sink Makes Request (Accept)”

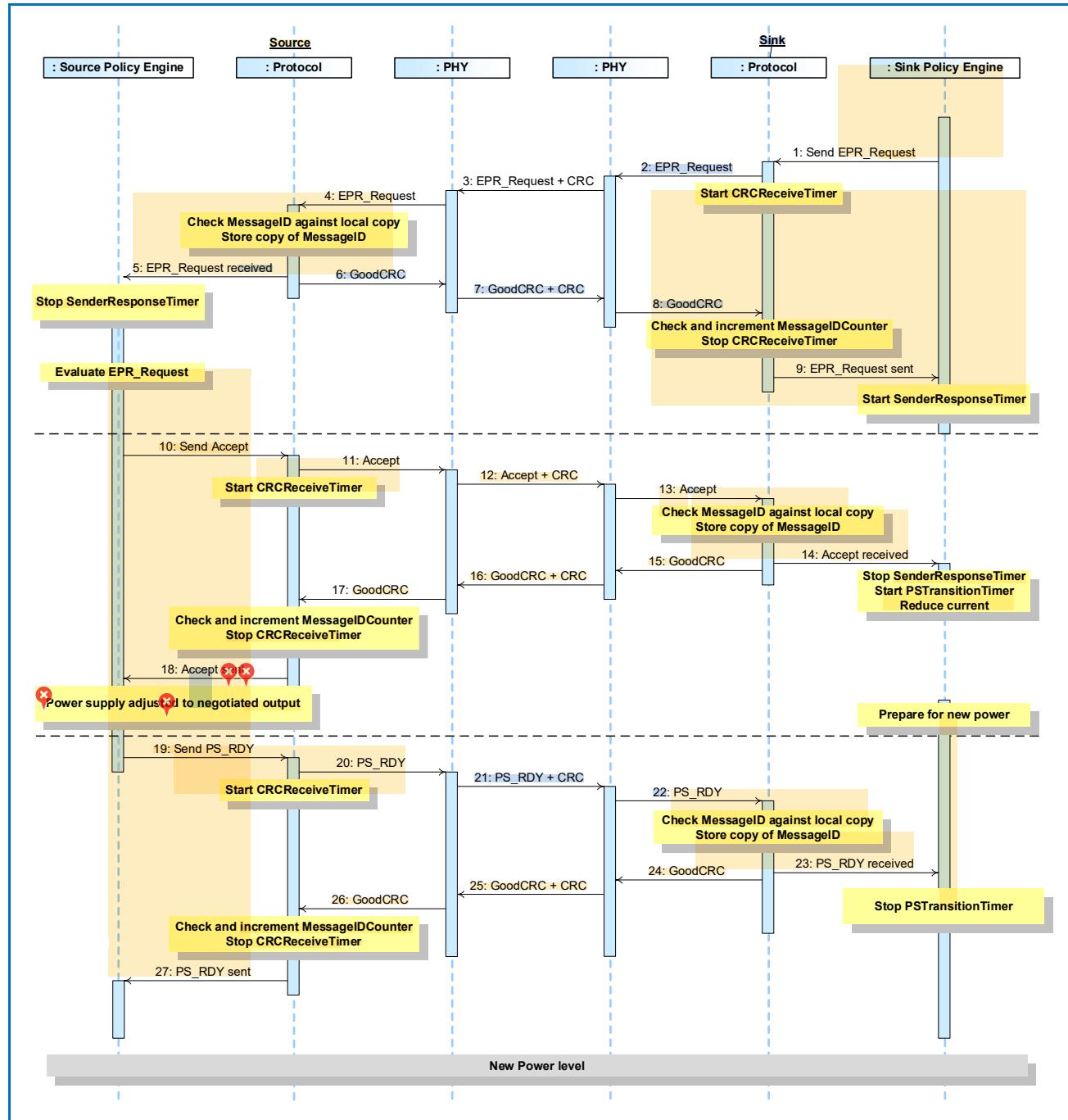


Table 8.50 “Steps for EPR Sink Makes Request (Accept)” below provides a detailed explanation of what happens at each labeled step in **Figure 8-22 “EPR Sink Makes Request (Accept)”** above.

Table 8.50 “Steps for EPR Sink Makes Request (Accept)”

Step	Source	Sink
1		DPM tells the Policy Engine to request a different power level. The Policy Engine tells the Protocol Layer to form an EPR_Request Message. The Protocol Layer creates the EPR_Request Message and passes it to Physical Layer.
2	Physical Layer receives the EPR_Request Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the EPR_Request Message. Starts CRCReceiveTimer .
3	Physical Layer removes the CRC and forwards the EPR_Request Message to the Protocol Layer.	
4	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine.	
5	The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.	
6	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
7		Physical Layer forwards the GoodCRC Message to the Protocol Layer.
8		The protocol Layer verifies and increments the MessageIDCounter . It informs the Policy Engine that the EPR_Request Message was successfully sent. The Protocol Layer stops the CRCReceiveTimer . The Policy Engine starts SenderResponseTimer .
9	Policy Engine requests the Device Policy Manager to evaluate the EPR_Request Message sent by the Sink and decides if the Source can meet the request. The Policy Engine tells the Protocol Layer to form an Accept Message.	
10	The Protocol Layer forms the Accept Message that is passed to the Physical Layer.	
11	Physical Layer appends CRC and sends the Accept Message. Starts CRCReceiveTimer .	Physical Layer receives the Accept Message and compares the CRC it calculated with the one sent to verify the Message.
12		Physical Layer forwards the Accept Message to the Protocol Layer.

13		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that an Accept Message has been received. The Policy Engine stops SenderResponseTimer , starts the PSTransitionTimer and reduces its current draw. The Device Policy Manager prepares the Power supply for transition to the new power level.
14		The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.
15	Physical Layer receives the GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
16	Physical Layer forwards the GoodCRC Message to the Protocol Layer. The Protocol Layer verifies and increments the MessageIDCounter and stops the CRCReceiveTimer .	
17	The Protocol Layer informs the Policy Engine that an Accept Message was successfully sent.	
Power supply Adjusts its Output to the Negotiated Value		
18	The Device Policy Manager informs the Policy Engine that the power supply has settled at the new operating condition and tells the Protocol Layer to send a PS_RDY Message.	
19	The Protocol Layer forms the PS_RDY Message.	
20	Physical Layer appends CRC and sends the PS_RDY Message. Starts CRCReceiveTimer .	Physical Layer receives the PS_RDY Message and compares the CRC it calculated with the one sent to verify the Message.
21		Physical Layer forwards the PS_RDY Message to the Protocol Layer.
22		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that a RS_RDY has been received. The Policy Engine stops the PSTransitionTimer .
23		The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.
24	Physical Layer receives the GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message. *	Physical Layer appends CRC and sends the GoodCRC Message.
25	Physical Layer forwards the GoodCRC Message to the Protocol Layer. The Protocol Layer verifies and increments the MessageIDCounter . Stops the CRCReceiveTimer .	
26	The Protocol Layer informs the Policy Engine that the PS_RDY Message was successfully sent.	
New Power Level Negotiated		

8.3.2.2.2.5.2

EPR Sink Makes Request (Reject)

This is an example of EPR when a Sink makes a Request which is Rejected during an Explicit Contract. [Figure 8-23 “EPR Sink Makes Request \(Reject\)”](#) shows the Messages as they flow across the bus and within the devices to accomplish the keep alive.

[Figure 8-23 “EPR Sink Makes Request \(Reject\)”](#)

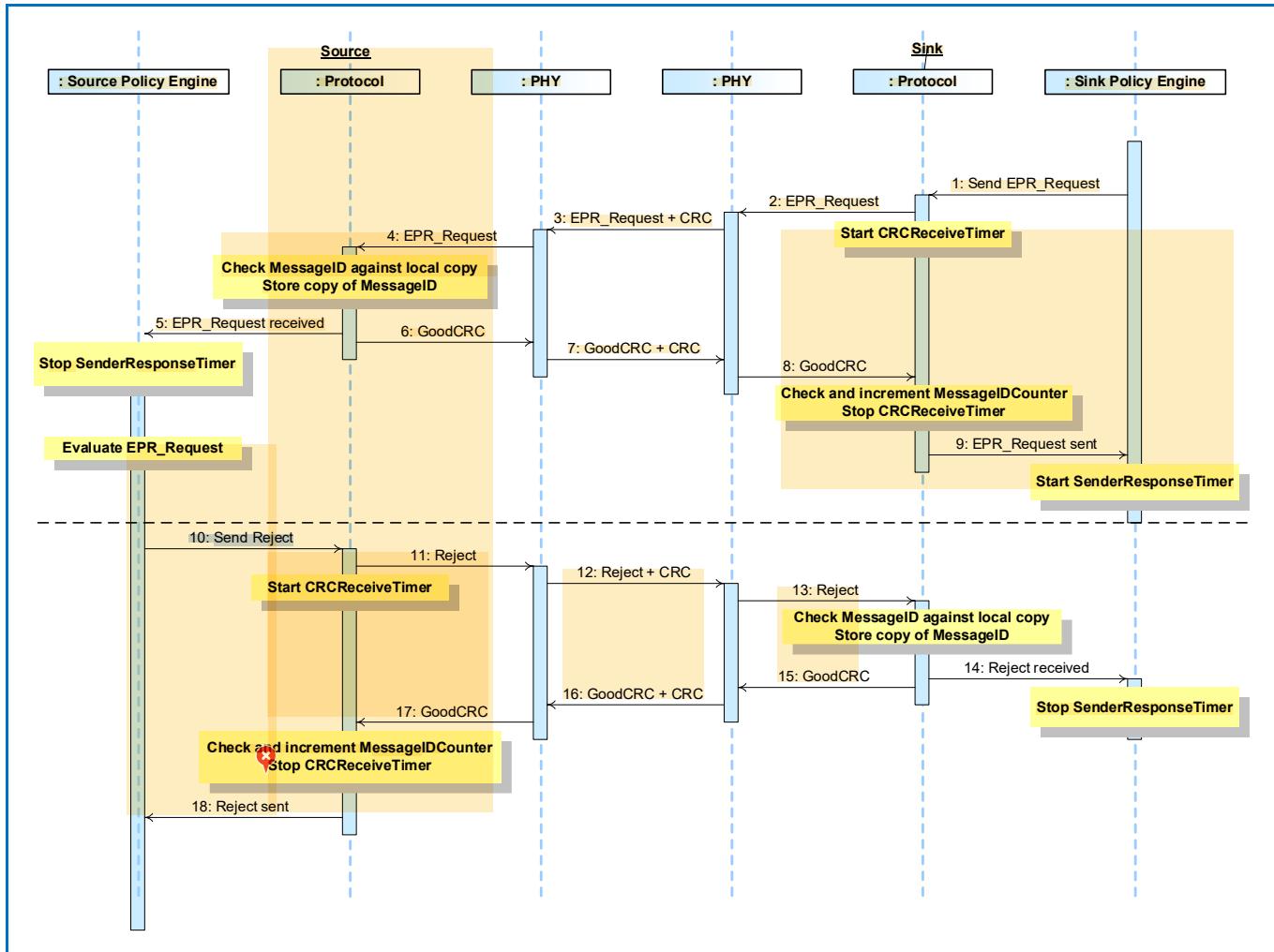


Table 8.51 “Steps for EPR Sink Makes Request (Reject)” below provides a detailed explanation of what happens at each labeled step in **Figure 8-23 “EPR Sink Makes Request (Reject)”** above.

Table 8.51 “Steps for EPR Sink Makes Request (Reject)”

Step	Source	Sink
1		DPM tells the Policy Engine to request a different power level. The Policy Engine tells the Protocol Layer to form an EPR_Request Message. The Protocol Layer creates the Request Message and passes it to Physical Layer.
2	Physical Layer receives the EPR_Request Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the EPR_Request Message. Starts CRCReceiveTimer .
3	Physical Layer removes the CRC and forwards the EPR_Request Message to the Protocol Layer.	
4	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine.	
5	The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.	
6	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
7		Physical Layer forwards the GoodCRC Message to the Protocol Layer.
8		The protocol Layer verifies and increments the MessageIDCounter . It informs the Policy Engine that the EPR_Request Message was successfully sent. The Protocol Layer stops the CRCReceiveTimer . The Policy Engine starts SenderResponseTimer .
9	Policy Engine requests the Device Policy Manager to evaluate the EPR_Request Message sent by the Sink and decides if the Source can meet the request. The Policy Engine tells the Protocol Layer to form a Reject Message.	
10	The Protocol Layer forms the Reject Message that is passed to the Physical Layer.	
11	Physical Layer appends CRC and sends the Reject Message. Starts CRCReceiveTimer .	Physical Layer receives the Reject Message and compares the CRC it calculated with the one sent to verify the Message.
12		Physical Layer forwards the Reject Message to the Protocol Layer.

13		<p>Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value.</p> <p>Protocol Layer informs the Policy Engine that an <i>Reject</i> Message has been received. The Policy Engine informs the Device Policy Manager that the Request has been rejected.</p>
14		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
15	Physical Layer receives the <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
16	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops the <i>CRCReceiveTimer</i> .	
17	The Protocol Layer informs the Policy Engine that a <i>Reject</i> Message was successfully sent.	

8.3.2.2.2.5.3

EPR Sink Makes Request (Wait)

This is an example of SPR when a Sink makes a Request which is responded to with Wait during an Explicit Contract. [Figure 8-24 “EPR Sink Makes Request \(Wait\)”](#) shows the Messages as they flow across the bus and within the devices to accomplish the keep alive.

[Figure 8-24 “EPR Sink Makes Request \(Wait\)”](#)

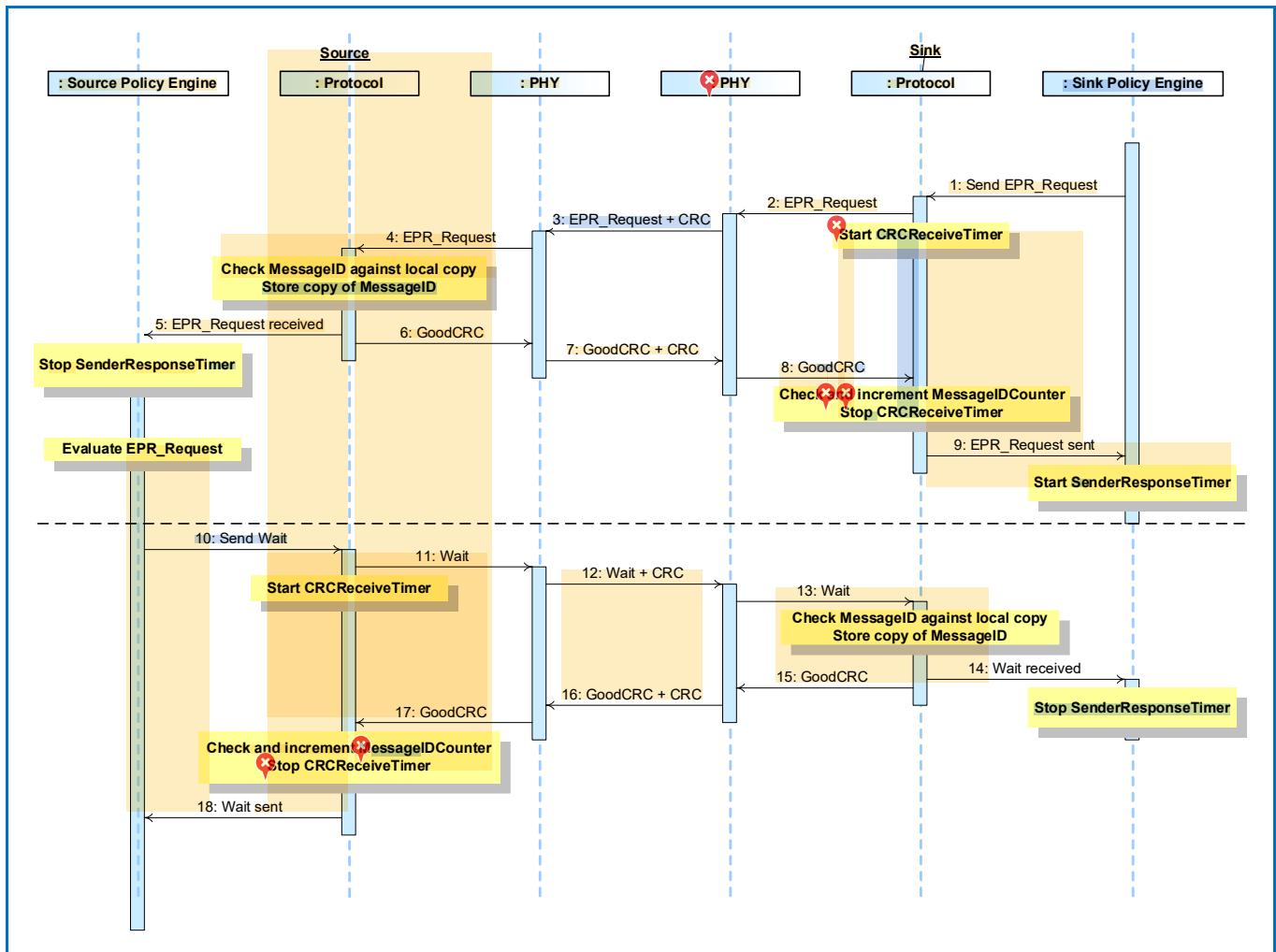


Table 8.40 "Steps for SPR Sink Makes Request (Wait)" below provides a detailed explanation of what happens at each labeled step in **Figure 8-24 "EPR Sink Makes Request (Wait)"** above.

Table 8.52 "Steps for EPR Sink Makes Request (Wait)"

Step	Source	Sink
1		DPM tells the Policy Engine to request a different power level. The Policy Engine tells the Protocol Layer to form an EPR_Request Message. The Protocol Layer creates the EPR_Request Message and passes it to Physical Layer.
2	Physical Layer receives the EPR_Request Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the EPR_Request Message. Starts CRCReceiveTimer .
3	Physical Layer removes the CRC and forwards the EPR_Request Message to the Protocol Layer.	
4	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer passes the Request information to the Policy Engine.	
5	The Protocol Layer generates a GoodCRC Message and passes it to its Physical Layer.	
6	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
7		Physical Layer forwards the GoodCRC Message to the Protocol Layer.
8		The protocol Layer verifies and increments the MessageIDCounter . It informs the Policy Engine that the Request Message was successfully sent. The Protocol Layer stops the CRCReceiveTimer . The Policy Engine starts SenderResponseTimer .
9	Policy Engine requests the Device Policy Manager to evaluate the EPR_Request Message sent by the Sink and decides if the Source can meet the request. The Policy Engine tells the Protocol Layer to form a Wait Message.	
10	The Protocol Layer forms the Wait Message that is passed to the Physical Layer.	
11	Physical Layer appends CRC and sends the Wait Message. Starts CRCReceiveTimer .	Physical Layer receives the Wait Message and compares the CRC it calculated with the one sent to verify the Message.
12		Physical Layer forwards the Wait Message to the Protocol Layer.

13		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. Protocol Layer informs the Policy Engine that an <i>Wait</i> Message has been received. The Policy Engine informs the Device Policy Manager that the Request has been rejected.
14		The Protocol Layer generates a <i>GoodCRC</i> Message and passes it to its Physical Layer.
15	Physical Layer receives the <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message. *	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
16	Physical Layer forwards the <i>GoodCRC</i> Message to the Protocol Layer. The Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops the <i>CRCReceiveTimer</i> .	
17	The Protocol Layer informs the Policy Engine that a <i>Wait</i> Message was successfully sent.	

8.3.2.3 Unsupported Message

This is an example of the response to an unsupported message. [Figure 8-25 “Unsupported message”](#) shows the Messages as they flow across the bus and within the devices.

[Figure 8-25 “Unsupported message”](#)

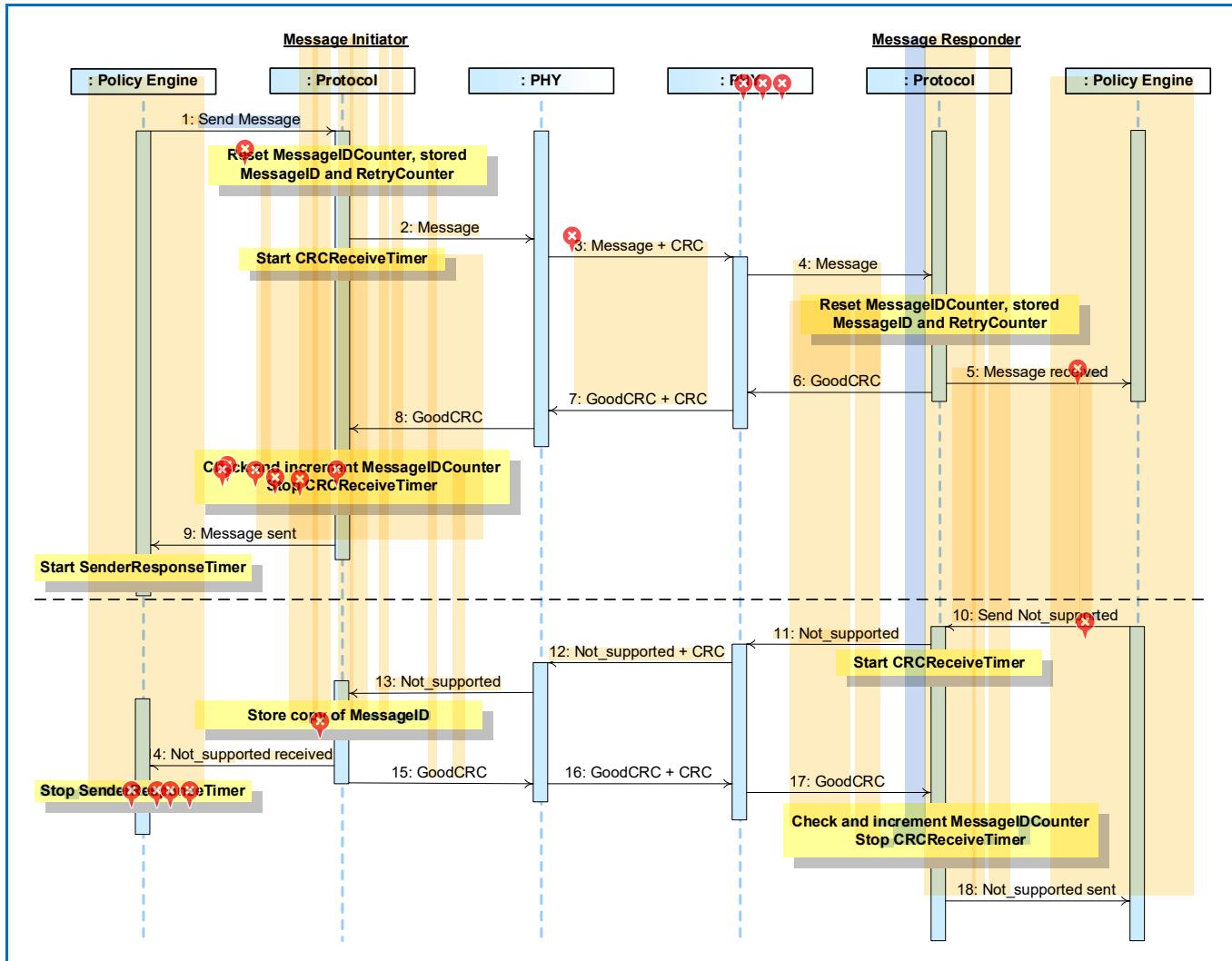


Table 8.53 “Steps for an Unsupported Message” below provides a detailed explanation of what happens at each labeled step in **Figure 8-25 “Unsupported message”** above.

Table 8.53 “Steps for an Unsupported Message”

Step	Message Initiator	Message Responder
1	The Policy Engine directs the Protocol Layer to generate a Message.	
2	Protocol Layer resets MessageIDCounter , stored MessageID and RetryCounter . Protocol Layer creates the Message and passes to Physical Layer.	
3 x	Physical Layer appends CRC and sends the Message. Starts CRCReceiveTimer .	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the Message to the Protocol Layer.
5 x		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the x GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine tells the Protocol Layer to form an Not_Supported Message.
11		Protocol Layer creates the Not_Supported Message and passes to Physical Layer.
12	Physical Layer receives the Not_Supported Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Not_Supported Message. Starts CRCReceiveTimer .
13	Protocol Layer stores the MessageID of the incoming Message.	
14	The Protocol Layer forwards the received Not_Supported Message information to the Policy Engine that consumes it.	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the x GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.

17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Not_Supported</i> Message was successfully sent.

8.3.2.4 Ping

This is an example of a ping sent from a Source to a Sink. [Figure 8-26 “Ping”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Soft Reset.

[Figure 8-26 “Ping”](#)

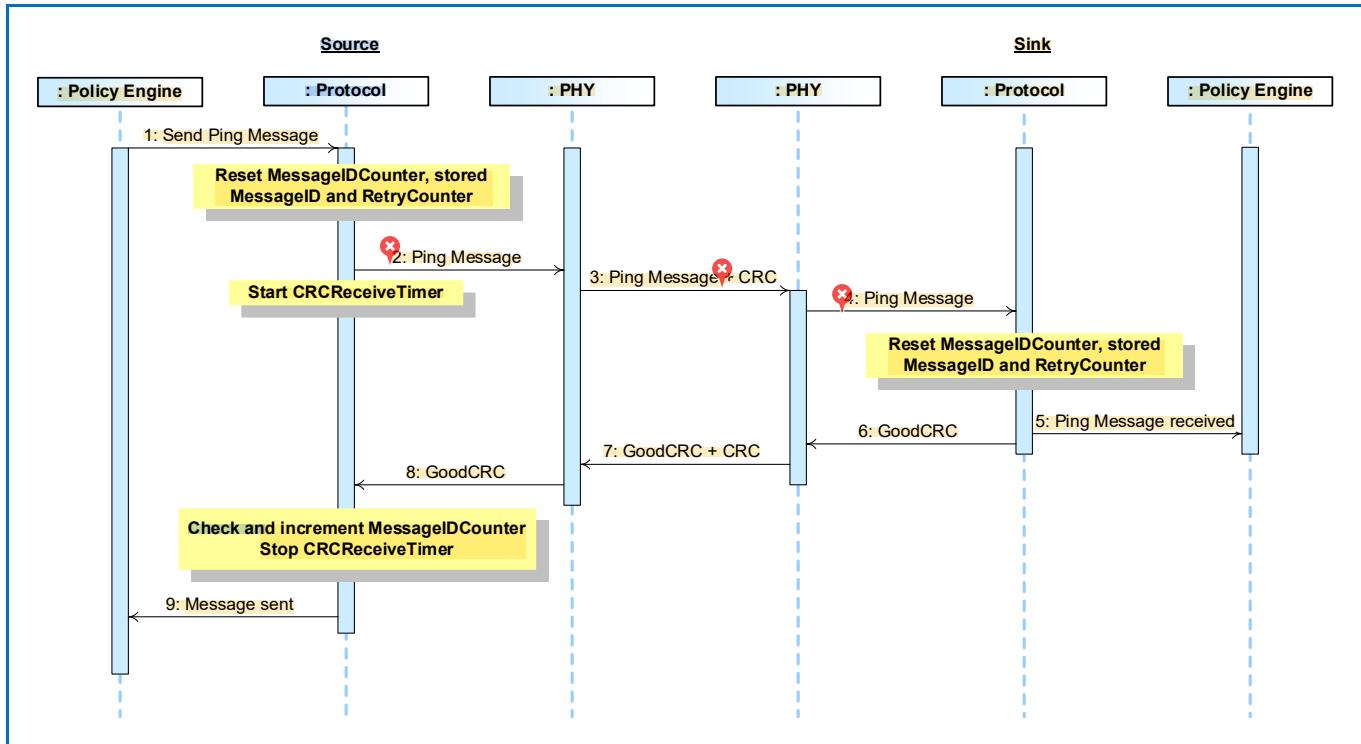


Table 8.54 “Steps for a Ping” below provides a detailed explanation of what happens at each labeled step in [Figure 8-26 “Ping”](#) above.

Table 8.54 “Steps for a Ping”

Step	Source	Sink
1	The Policy Engine directs the Protocol Layer to generate a Ping Message.	
2	Protocol Layer resets MessageIDCounter , stored MessageID and RetryCounter . Protocol Layer creates the Ping Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Ping Message. Starts CRCReceiveTimer .	Physical Layer receives the Ping Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the Ping Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received SoftPing Reset Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Message was successfully sent.	

8.3.2.5 Soft Reset

This is an example of a Soft Reset operation. [Figure 8-27 “Soft Reset”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Soft Reset.

[Figure 8-27 “Soft Reset”](#)

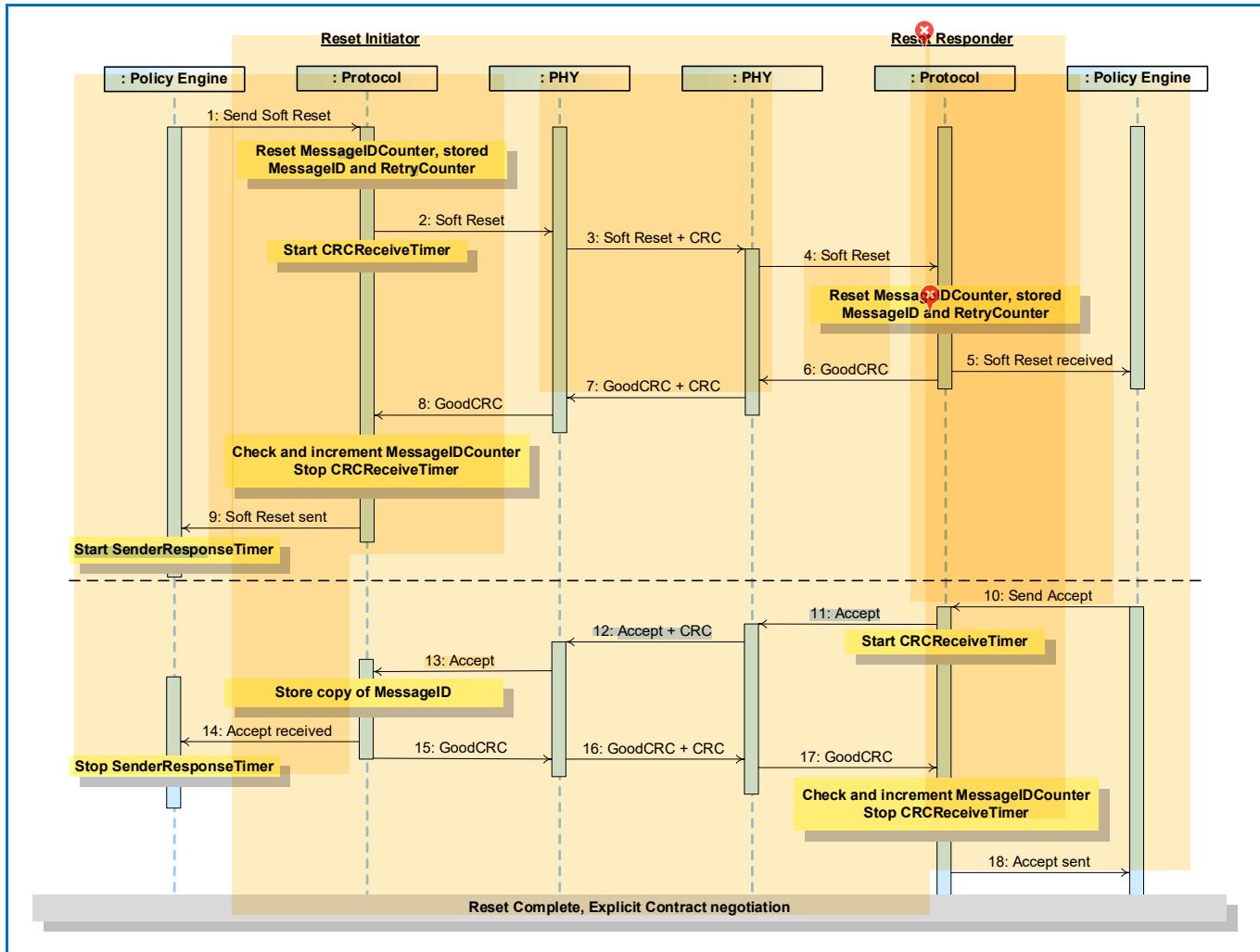


Table 8.55 “Steps for a Soft Reset” below provides a detailed explanation of what happens at each labeled step in **Figure 8-27 “Soft Reset”** above.

Table 8.55 “Steps for a Soft Reset”

Step	Reset Initiator	Reset Responder
1	The Policy Engine directs the Protocol Layer to generate a Soft_Reset Message to request a Soft Reset.	
2	Protocol Layer resets MessageIDCounter , stored MessageID and RetryCounter . Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Soft_Reset Message. Starts CRCReceiveTimer .	Physical Layer receives the Soft_Reset Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the Soft_Reset Message to the Protocol Layer.
5		Protocol Layer does not check the MessageID in the incoming Message and resets MessageIDCounter , stored MessageID and RetryCounter . The Protocol Layer forwards the received Soft_Reset Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Soft_Reset Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine tells the Protocol Layer to form an Accept Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Accept Message. Starts CRCReceiveTimer .
13	Protocol Layer stores the MessageID of the incoming Message.	
14	The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it.	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	

16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent.
The reset is complete and protocol communication can restart. Port Partners perform an Explicit Contract negotiation to re-synchronize their state machines.		

8.3.2.6📍 Data Reset

8.3.2.6.1 DFP Initiated Data Reset where the DFP is the VCONN Source

This is an example of a Data Reset operation where the DFP is also the VCONN Source and initiates a Data Reset.

Figure 8-28 “DFP Initiated Data Reset where the DFP is the Vconn Source” shows the Messages as they flow across the bus and within the devices to accomplish the Data Reset.

Figure 8-28 “DFP Initiated Data Reset where the DFP is the Vconn Source”

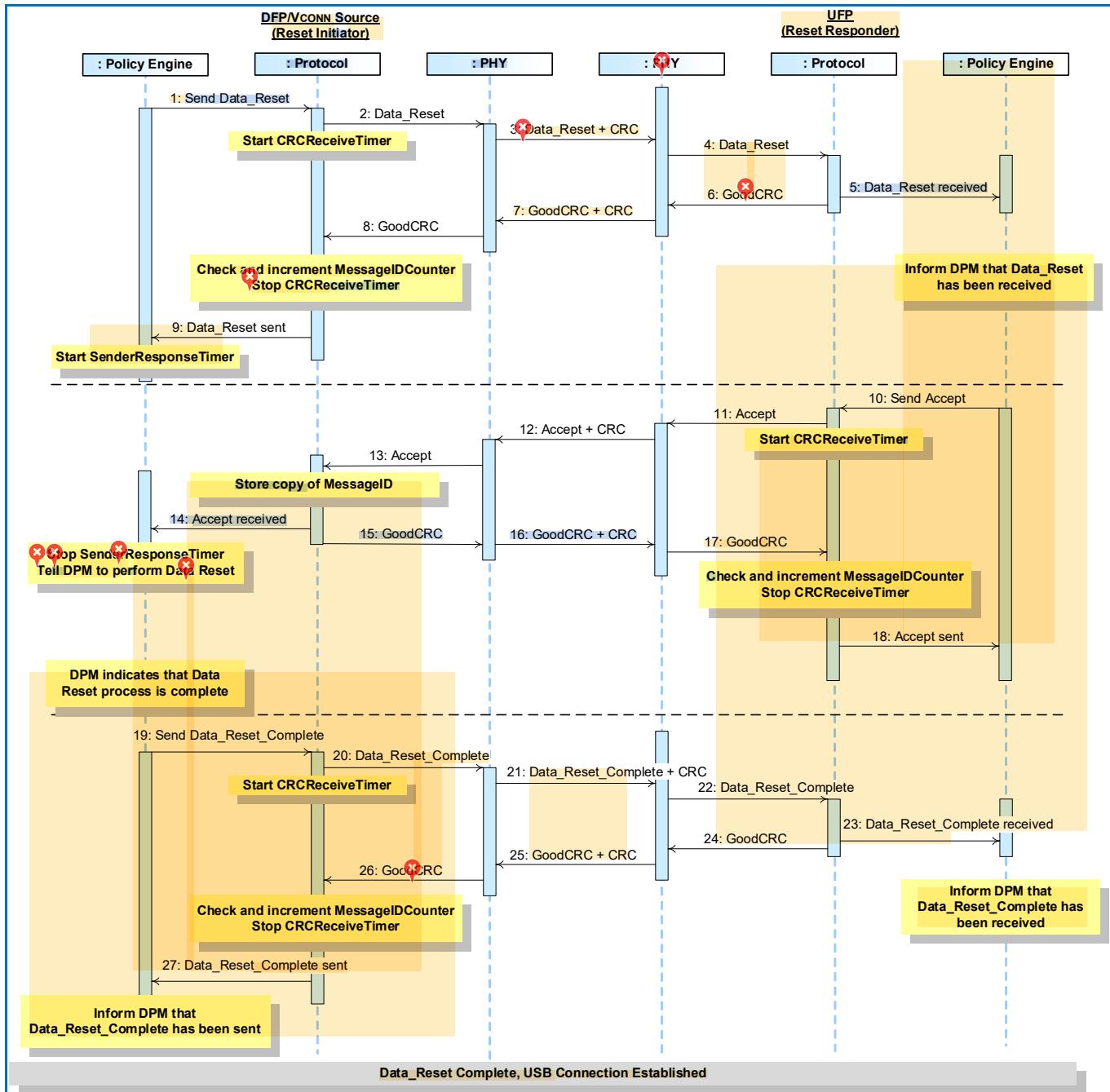


Table 8.56 “Steps for a DFP Initiated Data Reset where the DFP is the Vconn Source” below provides a detailed explanation of what happens at each labeled step in **Figure 8-28 “DFP Initiated Data Reset where the DFP is the Vconn Source”** above.

Table 8.56 “Steps for a DFP Initiated Data Reset where the DFP is the VCONN Source”

Step	DFP/VCONN Source (Reset Initiator)	UFP (Reset Responder)
1	The Policy Engine directs the Protocol Layer to generate a Data_Reset Message to request a Data Reset.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Data_Reset Message. Starts CRCReceiveTimer .	Physical Layer receives the Data_Reset Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the Data_Reset Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Data_Reset Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a Data_Reset Message has been received.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Data_Reset Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine tells the Protocol Layer to form an Accept Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Accept Message. Starts CRCReceiveTimer .
13	Protocol Layer stores the MessageID of the incoming Message.	

14	<p>The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it.</p> <p>The Policy Engine stops the SenderResponseTimer and tells the Device Policy Manager to perform a Data Reset.</p> <p>The Device Policy Manager proceeds to cycle VCONN and then reset the data connection.</p>	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
18		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Accept Message was successfully sent.
19	<p>The Device Policy Manager indicates that the Data Reset process is complete.</p> <p>The Policy Engine directs the Protocol Layer to generate a Data_Reset_Complete Message.</p>	
20	Protocol Layer creates the Message and passes to Physical Layer.	
21	Physical Layer appends CRC and sends the Data_Reset_Complete Message. Starts CRCReceiveTimer .	Physical Layer receives the Data_Reset_Complete Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer removes the CRC and forwards the Data_Reset_Complete Message to the Protocol Layer.
23		<p>Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value.</p> <p>The Protocol Layer forwards the received Data_Reset_Complete Message information to the Policy Engine that consumes it.</p> <p>The Policy Engine informs the Device Policy Manager that a Data_Reset_Complete Message has been received.</p>
24		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
25	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
26	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	

27	<p>Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i>.</p> <p>Protocol Layer informs the Policy Engine that the <i>Data_Reset_Complete</i> Message was successfully sent.</p> <p>The Policy Engine informs the Device Policy Manager that the <i>Data_Reset_Complete</i> Message was successfully sent.</p>	
<p>The data reset is complete as defined in Section 6.3.14 “Data_Reset Message” Step 5. Port Partners re-establish a USB data connection.</p>		

8.3.2.6.2

DFP Receives Data Reset where the DFP is the VCONN Source

This is an example of a Data Reset operation where the DFP receives a Data Reset Message and is the VCONN Source. [Figure 8-29 “DFP Receives Data Reset where the DFP is the Vconn Source”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Data Reset.

[Figure 8-29 “DFP Receives Data Reset where the DFP is the VCONN Source”](#)

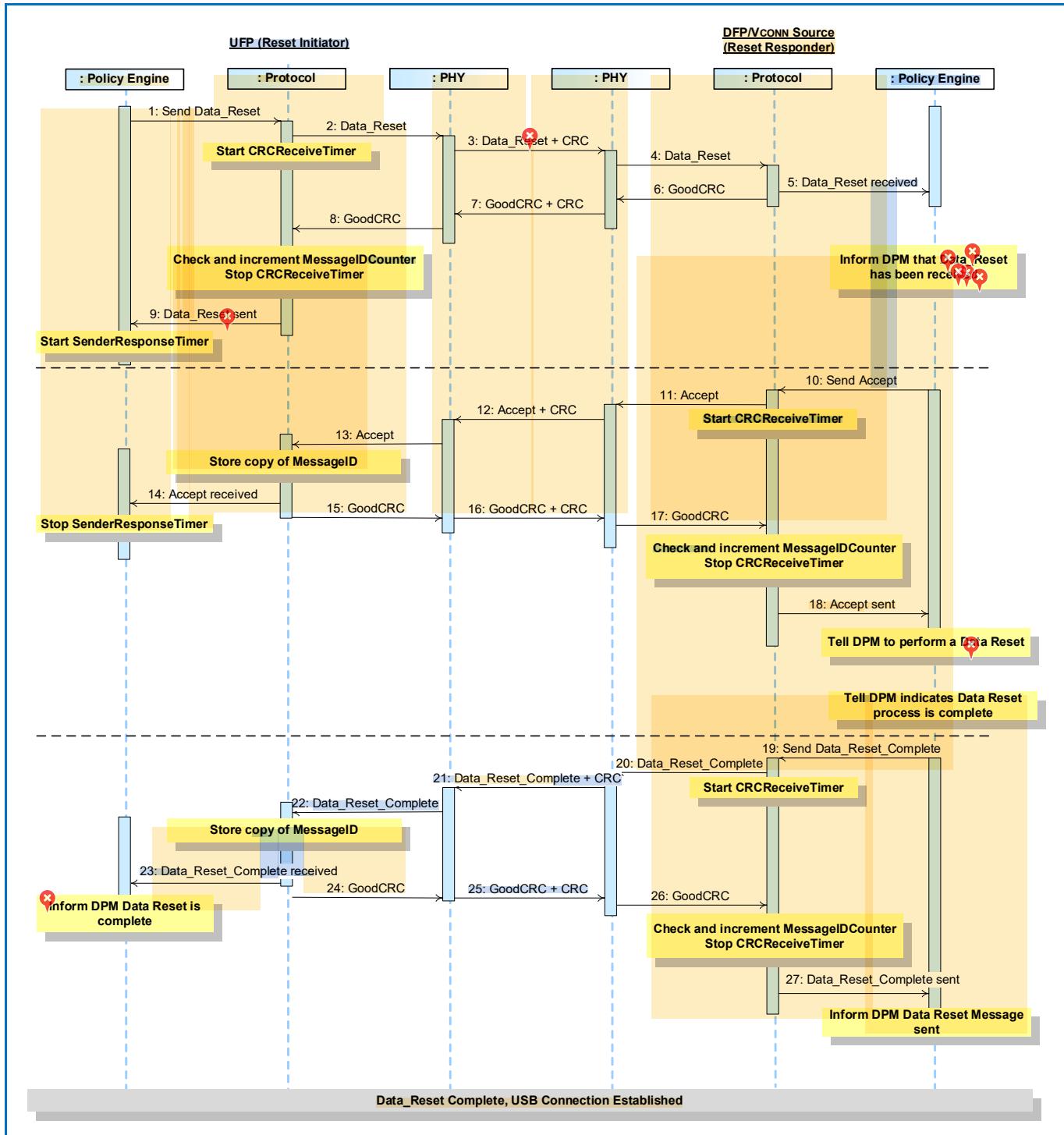


Table 8.57 “Steps for a DFP Receiving a Data Reset where the DFP is the Vconn Source” below provides a detailed explanation of what happens at each labeled step in **Figure 8-29 “DFP Receives Data Reset where the DFP is the Vconn Source”** above.

Table 8.57 “Steps for a DFP Receiving a Data Reset where the DFP is the VCONN Source”

Step	UFP (Reset Initiator)	DFP/VCONN Source (Reset Responder)
1	The Policy Engine directs the Protocol Layer to generate a Data_Reset Message to request a Data Reset.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Data_Reset Message. Starts CRCReceiveTimer .	Physical Layer receives the Data_Reset Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the Data_Reset Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Data_Reset Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a Data_Reset Message has been received.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Data_Reset Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine tells the Protocol Layer to form an Accept Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Message. Starts CRCReceiveTimer .
13	Protocol Layer stores the MessageID of the incoming Message.	

14	<p>The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it.</p> <p>The Policy Engine stops the SenderResponseTimer.</p> <p>The Device Policy Manager proceeds to cycle VCONN and then reset the data connection.</p>	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
18		<p>Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer.</p> <p>Protocol Layer informs the Policy Engine that the Accept Message was successfully sent.</p> <p>The Policy Engine tells the Device Policy Manager to perform a Data Reset.</p>
19		<p>The Device Policy Manager indicates that the Data Reset process is complete.</p> <p>The Policy Engine directs the Protocol Layer to generate a Data_Reset_Complete Message.</p>
20		Protocol Layer creates the Message and passes to Physical Layer.
21	Physical Layer receives the Data_Reset_Complete Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Data_Reset_Complete Message. Starts CRCReceiveTimer .
22	Physical Layer removes the CRC and forwards the Data_Reset_Complete Message to the Protocol Layer.	
23	<p>Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value.</p> <p>The Protocol Layer forwards the received Data_Reset_Complete Message information to the Policy Engine that consumes it.</p> <p>The Policy Engine informs the Device Policy Manager that a Data_Reset_Complete Message has been received.</p>	
24	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
25	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.
26		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.

27	<p>Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i>. Protocol Layer informs the Policy Engine that the <i>Data_Reset_Complete</i> Message was successfully sent.</p> <p>The Policy Engine informs the Device Policy Manager that the <i>Data_Reset_Complete</i> Message was successfully sent.</p>
<p>The reset is complete as defined in Section 6.3.14 "Data_Reset Message" Step 5 Port Partners re-establish a USB data connection.</p>	

8.3.2.6.3

DFP Initiated Data Reset where the UFP is the VCONN Source

This is an example of a Data Reset operation where the DFP initiates a Data Reset and the UFP is the VCONN Source.

Figure 8-30 “DFP Initiated Data Reset where the UFP is the Vconn Source” shows the Messages as they flow across the bus and within the devices to accomplish the Data Reset.

Figure 8-30 “DFP Initiated Data Reset where the UFP is the Vconn Source”

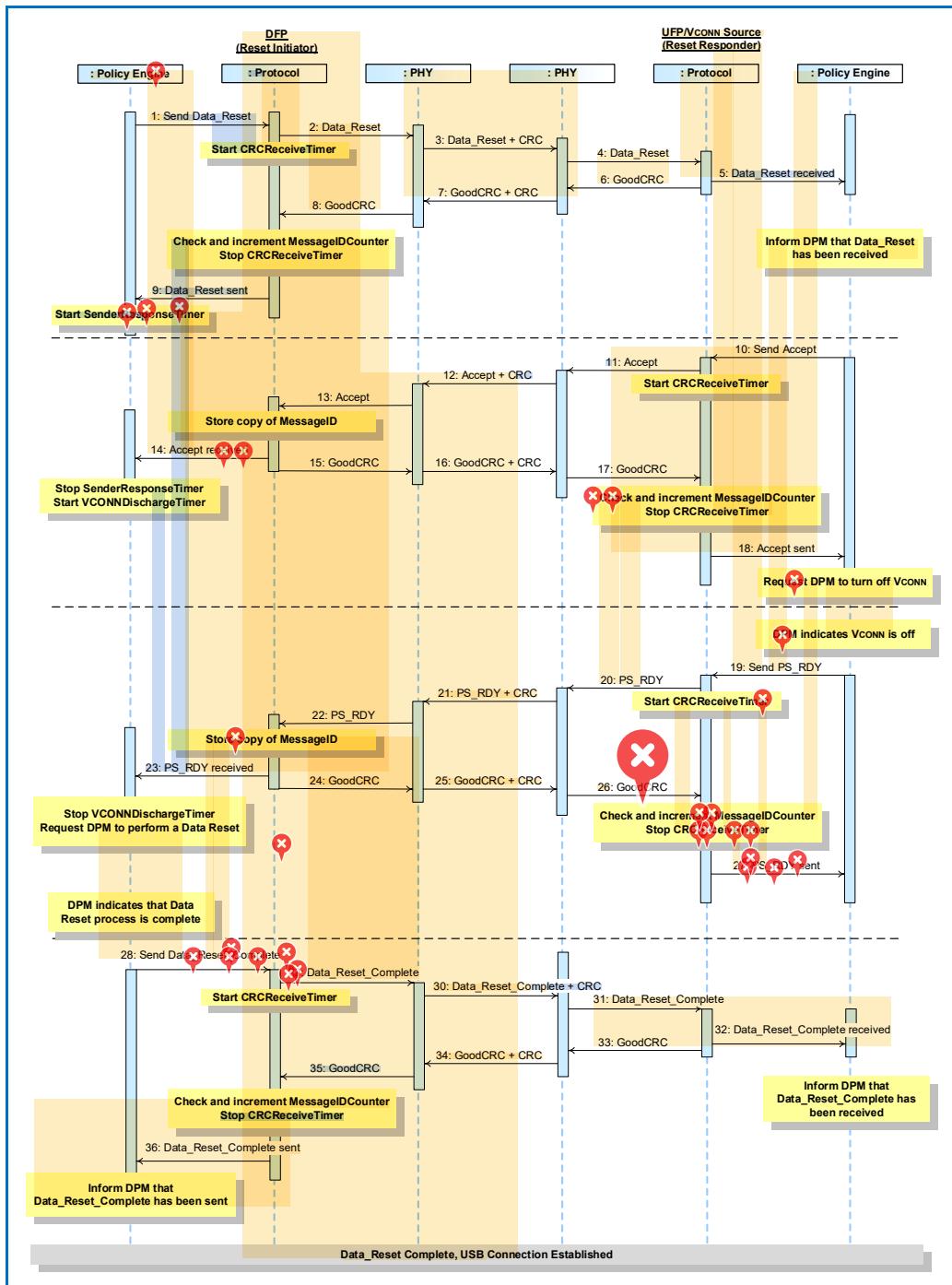


Table 8.58 “Steps for a DFP Initiated Data Reset where the UFP is the Vconn Source” below provides a detailed explanation of what happens at each labeled step in **Figure 8-30 “DFP Initiated Data Reset where the UFP is the Vconn Source”** above.

Table 8.58 “Steps for a DFP Initiated Data Reset where the UFP is the VCONN Source”

Step	DFP (Reset Initiator)	UFP/VCONN Source (Reset Responder)
1	The Policy Engine directs the Protocol Layer to generate a Data_Reset Message to request a Soft Reset.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Data_Reset Message. Starts CRCReceiveTimer .	Physical Layer receives the Data_Reset Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the Data_Reset Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Data_Reset Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a Data_Reset Message has been received.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Data_Reset Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine tells the Protocol Layer to form an Accept Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Message. Starts CRCReceiveTimer .
13	Protocol Layer stores the MessageID of the incoming Message.	
14	The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it. The Policy Engine stops the SenderResponseTimer and starts the VCONNDischargeTimer .	

15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer. 
18		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Accept Message was successfully sent. The Policy Engine requests the Device Policy Manager to turn off VCONN.
19		When the Device Policy Manager indicates VCONN has been turned off the Policy Engine tells the Protocol Layer to form an PS_RDY Message.
20		Protocol Layer creates the Message and passes to Physical Layer.
21	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Accept Message. Starts CRCReceiveTimer .
22	Protocol Layer stores the MessageID of the incoming Message.	
23	The Protocol Layer forwards the received PS_RDY Message information to the Policy Engine that consumes it. The Policy Engine stops the VCONNDischargeTimer and tells the Device Policy Manager to perform a Data Reset. The Device Policy Manager proceeds to turn on VCONN and then reset the data connection.	
24	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
25	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
26		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
27		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the PS_RDY Message was successfully sent.
28	The Device Policy Manager indicates that the Data Reset process is complete. The Policy Engine directs the Protocol Layer to generate a Data_Reset_Complete Message.	
29	Protocol Layer creates the Message and passes to Physical Layer.	

30	Physical Layer appends CRC and sends the Data_Reset_Complete Message. Starts CRCReceiveTimer .	Physical Layer receives the Data_Reset_Complete Message and compares the CRC it calculated with the one sent to verify the Message.
31		Physical Layer removes the CRC and forwards the Data_Reset_Complete Message to the Protocol Layer.
32		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Data_Reset_Complete Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a Data_Reset_Complete Message has been received.
33		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
34	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
35	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
36	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Data_Reset_Complete Message was successfully sent. The Policy Engine informs the Device Policy Manager that the Data_Reset_Complete Message was successfully sent.	
The reset is complete as defined in Section 6.3.14 "Data Reset Message" Step 5. Port Partners re-establish a USB data connection.		

8.3.2.6.4

DFP Receives Data Reset where the UFP is the VCONN Source

This is an example of a Data Reset operation where the DFP receives a Data Reset Message and the UFP is the VCONN Source. [Figure 8-31 “DFP Receives a Data Reset where the UFP is the Vconn Source”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Data Reset.

Figure 8-31 “DFP Receives a Data Reset where the UFP is the VCONN Source”

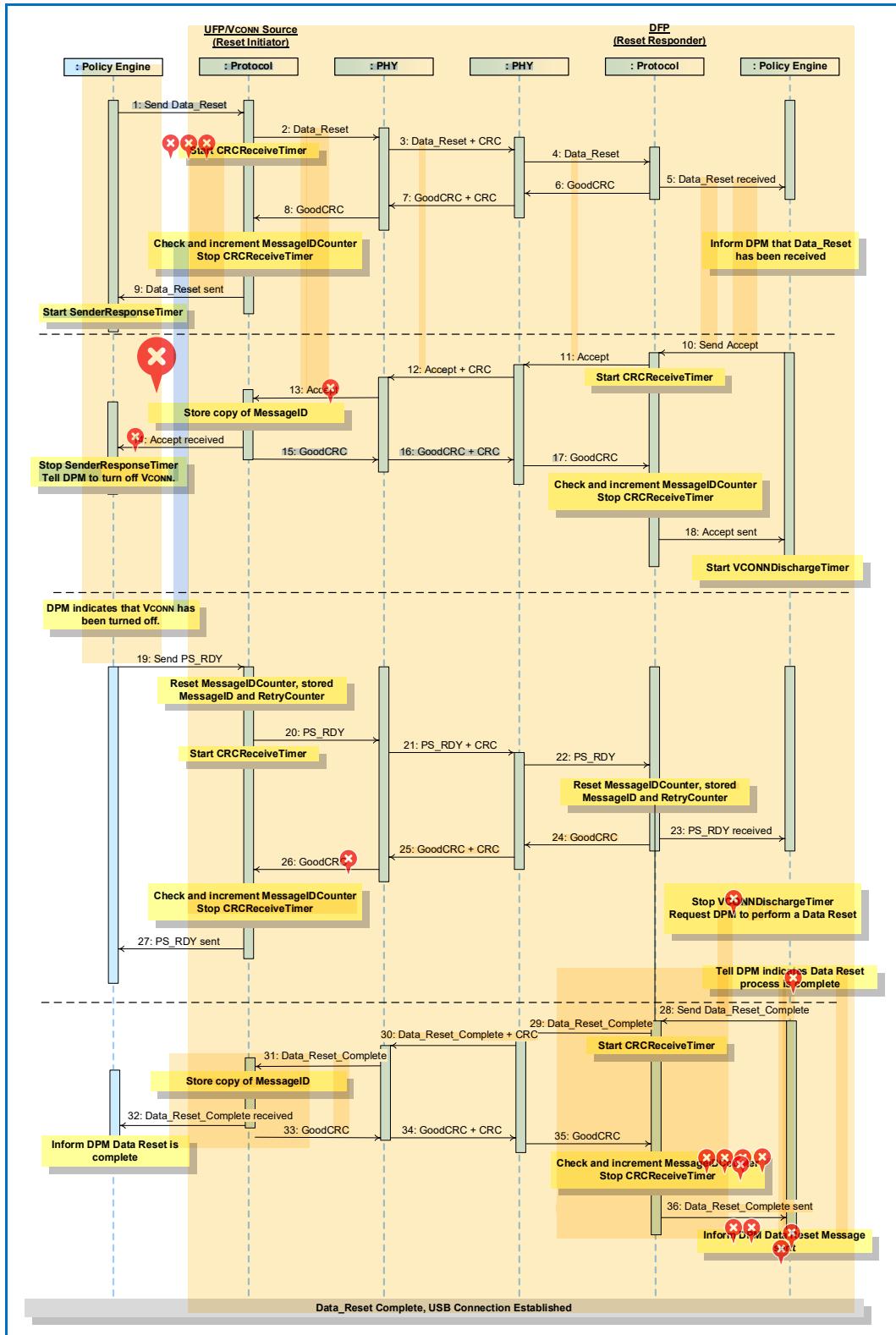


Table 8.59 "Steps for a DFP Receiving a Data Reset where the UFP is the Vconn Source" below provides a detailed explanation of what happens at each labeled step in **Figure 8-31 "DFP Receives a Data Reset where the UFP is the Vconn Source"** above.

Table 8.59 "Steps for a DFP Receiving a Data Reset where the UFP is the VCONN Source"

Step	UFP/VCONN Source (Reset Initiator)	DFP (Reset Responder)
1	The Policy Engine directs the Protocol Layer to generate a Data_Reset Message to request a Soft Reset.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Data_Reset Message. Starts CRCReceiveTimer .	Physical Layer receives the Data_Reset Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the Data_Reset Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Data_Reset Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a Data_Reset Message has been received.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Data_Reset Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine tells the Protocol Layer to form an Accept Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Message. Starts CRCReceiveTimer .
13	Protocol Layer stores the MessageID of the incoming Message.	
14	The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it. The Policy Engine stops the SenderResponseTimer and tells the Device Policy Manager to turn off VCONN.	



15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine starts the <i>VCONNDischargeTimer</i> .
19	When the Device Policy Manager indicates that VCONN has been turned off the Policy Engine directs the Protocol Layer to generate a <i>PS_RDY</i> Message to request a Soft Reset.	
20	Protocol Layer creates the Message and passes to Physical Layer.	
21	Physical Layer appends CRC and sends the <i>PS_RDY</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the <i>PS_RDY</i> Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.
23		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>VCONNDischargeTimer</i> and requests the Device Policy Manager perform a Data Reset. The Device Policy Manager proceeds to turn on VCONN and then reset the data connection.
24		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
25	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
26	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
27	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent.	
28		The Device Policy Manager indicates that the Data Reset process is complete. The Policy Engine directs the Protocol Layer to generate a <i>Data_Reset_Complete</i> Message.

29		Protocol Layer creates the Message and passes to Physical Layer.
30	Physical Layer receives the Data_Reset_Complete Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends CRC and sends the Data_Reset_Complete Message. Starts CRCReceiveTimer .
31	Physical Layer removes the CRC and forwards the Data_Reset_Complete Message to the Protocol Layer.	
32	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Data_Reset_Complete Message information to the Policy Engine that consumes it. The Policy Engine informs the Device Policy Manager that a Data_Reset_Complete Message has been received.	
33	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
34	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.
35		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
36		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Data_Reset_Complete Message was successfully sent. The Policy Engine informs the Device Policy Manager that the Data_Reset_Complete Message was successfully sent.
The reset is complete as defined in Section 6.3.14 "Data_Reset Message" Step 5. Port Partners re-establish a USB data connection.		

8.3.2.7 Hard Reset

 The following sections describe the steps required for a USB Power Delivery Hard Reset. The Hard Reset returns the operation of the USB Power Delivery to default role and operating Voltage/current. During the Hard-Reset USB Power Delivery PHY Layer communications ***Shall*** be disabled preventing communication between the Port partners.

Note: Hard Reset, in this case, is applied to the USB Power Delivery capability of an individual Port on which the Hard Reset is requested. A side effect of the Hard Reset is that it might reset other functions on the Port such as USB.

8.3.2.7.1 Source Initiated Hard Reset

This is an example of a Hard-Reset operation when initiated by a Source. [**Figure 8-32 “Source initiated Hard Reset”**](#) shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.

Figure 8-32 "Source initiated Hard Reset"

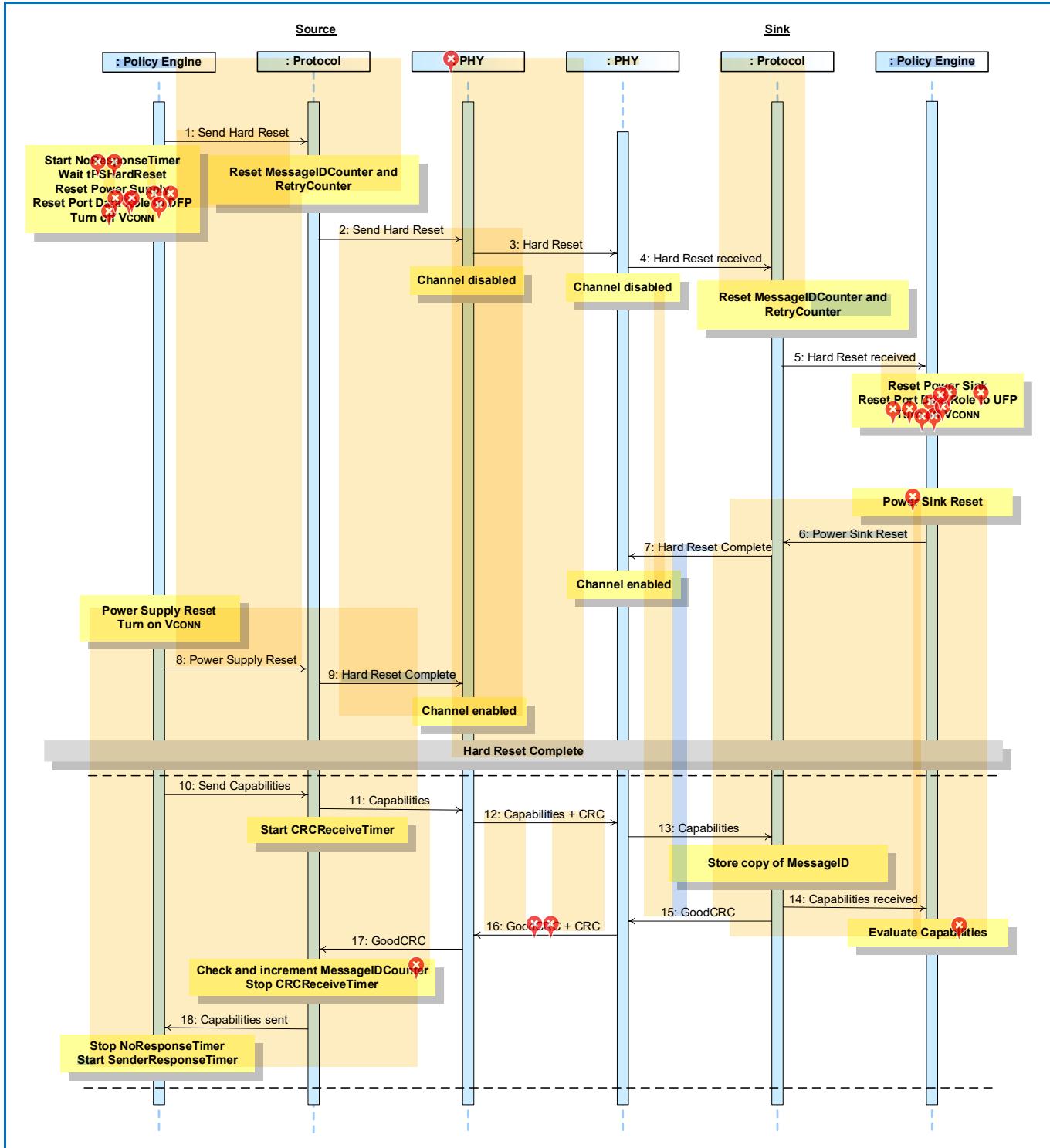


Table 8.60 “Steps for Source initiated Hard Reset” below provides a detailed explanation of what happens at each labeled step in **Figure 8-32 “Source initiated Hard Reset”** above.

Table 8.60 “Steps for Source initiated Hard Reset”

Step	Source	Sink
1	The Policy Engine directs the Protocol Layer to generate Hard Reset Signaling. The Policy Engine starts the NoResponseTimer and requests the Device Policy Manager to reset the power supply to USB Default Operation. The Policy Engine requests the Device Policy Manager to reset the Port Data Role to DFP and to turn off VCONN if this is on.	
2	Protocol Layer resets MessageIDCounter and RetryCounter . Protocol Layer requests the Physical Layer send Hard Reset Signaling.	
3	Physical Layer sends Hard Reset Signaling and then disables the PHY Layer communications channel for transmission and reception.	Physical Layer receives the Hard Reset Signaling and disables the PHY Layer communications channel for transmission and reception.
4		Physical Layer informs the Protocol Layer of the Hard Reset. Protocol Layer resets MessageIDCounter and RetryCounter .
5		The Protocol Layer informs the Policy Engine of the Hard Reset. The Policy Engine requests the Device Policy Manager to reset the Power Sink to USB Default Operation. The Policy Engine requests the Device Policy Manager to reset the Port Data Role to UFP and to turn off VCONN if this is on.
6		The Power Sink returns to USB Default Operation.. The Policy Engine informs the Protocol Layer that the Power Sink has been reset.
7		The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.
8	The power supply is reset to USB Default Operation. and VCONN is turned on. The Policy Engine informs the Protocol Layer that the power supply has been reset.	
9	The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.	
	The reset is complete and protocol communication can restart.	
10	Policy Engine directs the Protocol Layer to send a Source_Capabilities Message that represents the power supply's present capabilities.	

11	Protocol Layer creates the Message and passes to Physical Layer.	
12	Physical Layer appends CRC and sends the Source_Capabilities Message. Starts CRCReceiveTimer .	Physical Layer receives the Source_Capabilities Message and checks the CRC to verify the Message.
13		Physical Layer removes the CRC and forwards the Source_Capabilities Message to the Protocol Layer.
14		Protocol Layer stores the MessageID of the incoming Message. The Protocol Layer forwards the received Source_Capabilities Message information to the Policy Engine that consumes it.
15		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
16	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
17	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Source_Capabilities Message was successfully sent. Policy Engine stops the NoResponseTimer and starts the SenderResponseTimer .	
USB Power Delivery communication is re-established.		

8.3.2.7.2 Sink Initiated Hard Reset

This is an example of a Hard-Reset operation when initiated by a Sink. [Figure 8-33 “Sink Initiated Hard Reset”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.

[Figure 8-33 “Sink Initiated Hard Reset”](#)

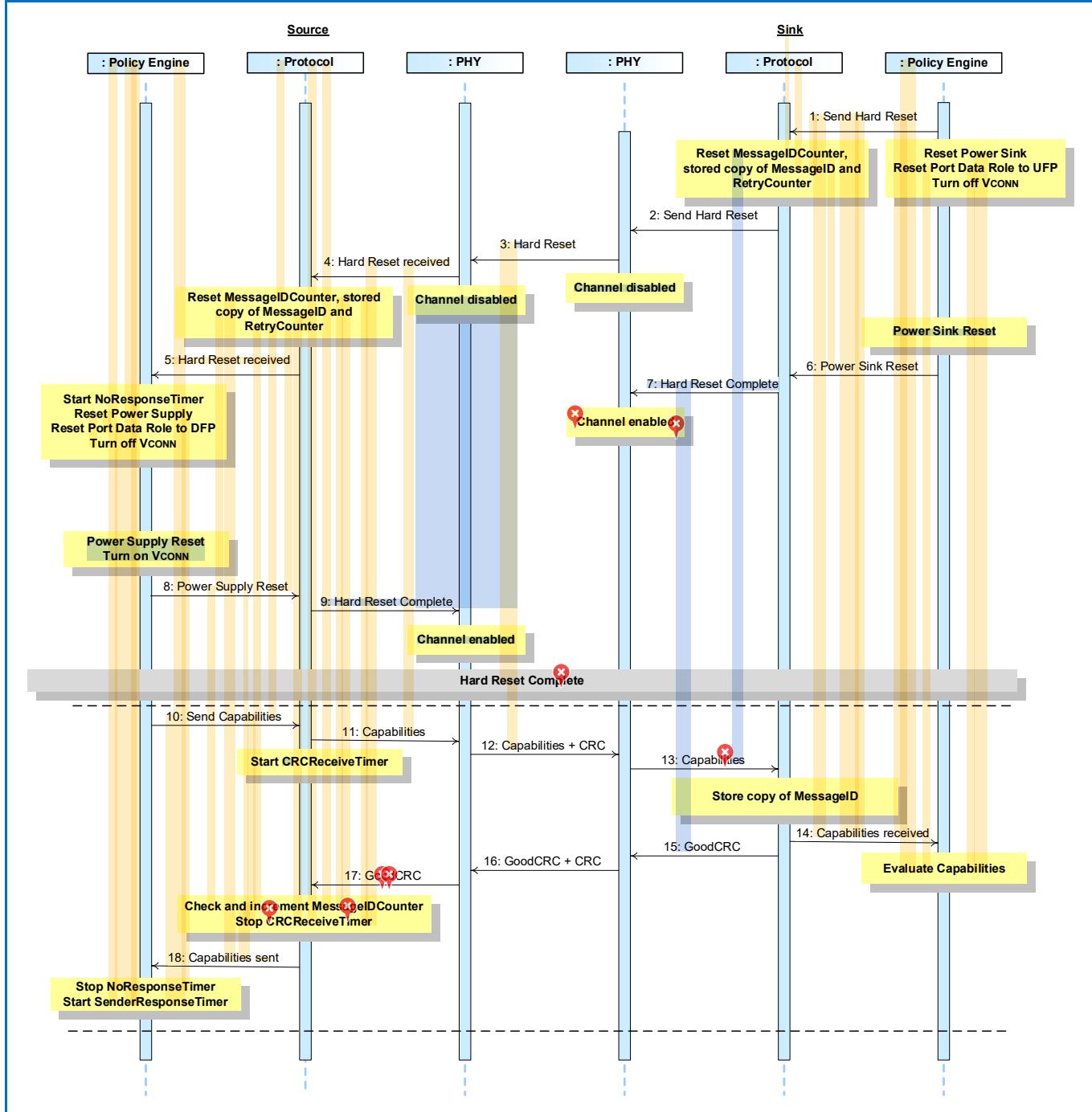


Table 8.61 “Steps for Sink initiated Hard Reset” below provides a detailed explanation of what happens at each labeled step in **Figure 8-33 “Sink Initiated Hard Reset”** above.

Table 8.61 “Steps for Sink initiated Hard Reset”

Step	Source	Sink
1		The Policy Engine directs the Protocol Layer to generate Hard Reset Signaling. The Policy Engine requests the Device Policy Manager to reset the power supply to USB Default Operation. The Policy Engine requests the Device Policy Manager to reset the Port Data Role to UFP and to turn off VCONN if this is on.
2		Protocol Layer resets MessageIDCounter , stored copy of MessageID and RetryCounter . Protocol Layer requests the Physical Layer send Hard Reset Signaling.
3	Physical Layer receives the Hard Reset Signaling and disables the PHY Layer communications channel for transmission and reception.	Physical Layer sends the Hard Reset Signaling and then disables the PHY Layer communications channel for transmission and reception.
4	Physical Layer informs the Protocol Layer of the Hard Reset. Protocol Layer resets MessageIDCounter , stored copy of MessageID and RetryCounter .	
5	The Protocol Layer Informs the Policy Engine of the Hard Reset. The Policy Engine starts the NoResponseTimer and requests the Device Policy Manager to reset the Power Sink to USB Default Operation . The Policy Engine requests the Device Policy Manager to reset the Port Data Role to DFP and to turn off VCONN if this is on.	
6		The Power Sink returns to USB Default Operation. The Policy Engine informs the Protocol Layer that the Power Sink has been reset.
7		The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.
8	The power supply is reset to USB Default Operation and VCONN is turned on. The Policy Engine informs the Protocol Layer that the power supply has been reset.	
9	The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.	
	The reset is complete and protocol communication can restart.	
10	Policy Engine directs the Protocol Layer to send a Source_Capabilities Message that represents the power supply's present capabilities.	

11	Protocol Layer creates the Message and passes to Physical Layer.	
12	Physical Layer appends CRC and sends the Source_Capabilities Message. Starts CRCReceiveTimer .	Physical Layer receives the Source_Capabilities Message and checks the CRC to verify the Message.
13		Physical Layer removes the CRC and forwards the Source_Capabilities Message to the Protocol Layer.
14		Protocol Layer stores the MessageID of the incoming Message. The Protocol Layer forwards the received Source_Capabilities Message information to the Policy Engine that consumes it.
15		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
16	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
17	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Source_Capabilities Message was successfully sent. Policy Engine stops the NoResponseTimer and starts the SenderResponseTimer .	
USB Power Delivery communication is re-established.		

8.3.2.7.3

Source Initiated Hard Reset - Sink Long Reset

This is an example of a Hard-Reset operation when initiated by a Source. In this example the Sink is slow responding to the reset causing the Source to send multiple **Source_Capabilities** Messages before it receives a **GoodCRC** Message response. [Figure 8-34 “Source initiated reset - Sink long reset”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Hard Reset.

[Figure 8-34 “Source initiated reset - Sink long reset”](#)

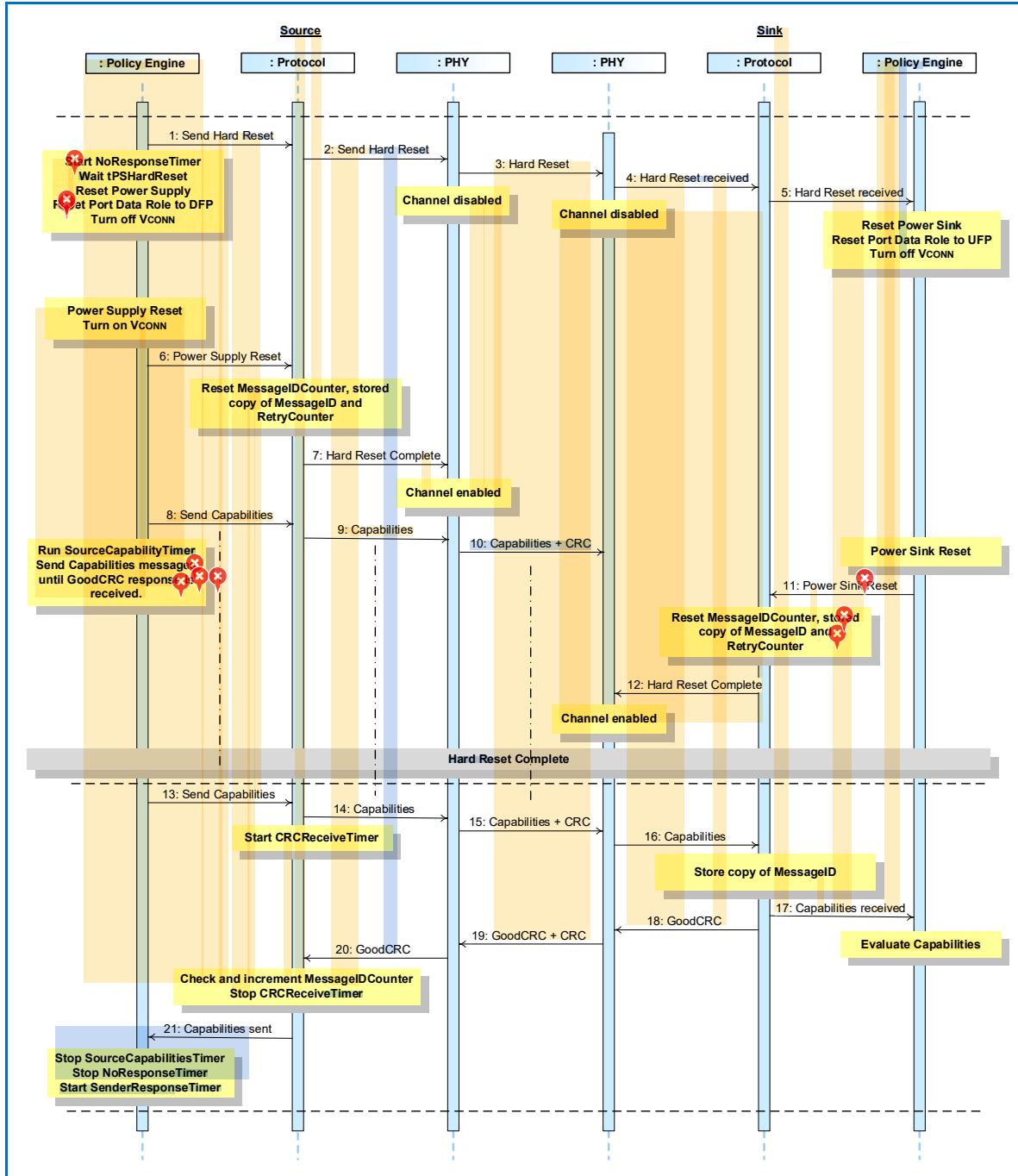


Table 8.62 “Steps for Source initiated Hard Reset – Sink long reset” below provides a detailed explanation of what happens at each labeled step in **Figure 8-34 “Source initiated reset - Sink long reset”** above.

Table 8.62 “Steps for Source initiated Hard Reset – Sink long reset”

Step	Source	Sink
1	The Policy Engine directs the Protocol Layer to generate Hard Reset Signaling. The Policy Engine starts the NoResponseTimer and requests the Device Policy Manager to reset the power supply to USB Default Operation. The Policy Engine requests the Device Policy Manager to reset the Port Data Role to DFP and to turn off VCONN if this is on.	
2	Protocol Layer resets MessageIDCounter , stored copy of MessageID and RetryCounter . Protocol Layer requests the Physical Layer send Hard Reset Signaling.	
3	Physical Layer sends the Hard Reset Signaling and then disables the PHY Layer communications channel for transmission and reception.	Physical Layer receives the Hard Reset Signaling and disables the PHY Layer communications channel for transmission and reception.
4		Physical Layer informs the Protocol Layer of the Hard Reset. Protocol Layer resets MessageIDCounter , stored copy of MessageID and RetryCounter .
5		The Protocol Layer Informs the Policy Engine of the Hard Reset. The Policy Engine requests the Device Policy Manager to reset the Power Sink to USB Default Operation.. The Policy Engine requests the Device Policy Manager to reset the Port Data Role to UFP and to turn off VCONN if this is on.
6	The power supply is reset to USB Default Operation and VCONN is turned on. The Policy Engine informs the Protocol Layer that the power supply has been reset.	
7	The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.	
	The reset is complete and protocol communication can restart.	
8	Policy Engine directs the Protocol Layer to send a Source_Capabilities Message that represents the power supply's present capabilities. Policy Engine starts the SourceCapabilityTimer . The SourceCapabilityTimer times out one or more times until a GoodCRC Message response is received.	
9	Protocol Layer creates the Message and passes to Physical Layer.	

10	Physical Layer appends CRC and sends the <i>Source_Capabilities</i> Message. Starts <i>CRCReceiveTimer</i> .	Note: <i>Source_Capabilities</i> Message not received since channel is disabled.
11		The Power Sink returns to USB Default Operation. The Policy Engine informs the Protocol Layer that the Power Sink has been reset.
12		The Protocol Layer informs the PHY Layer that the Hard Reset is complete. The PHY Layer enables the PHY Layer communications channel for transmission and reception.
The reset is complete and protocol communication can restart.		
13	Policy Engine directs the Protocol Layer to send a <i>Source_Capabilities</i> Message that represents the power supply's present capabilities. Starts the <i>SourceCapabilityTimer</i> .	
14	Protocol Layer creates the Message and passes to Physical Layer.	
15	Physical Layer appends CRC and sends the <i>Source_Capabilities</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the <i>Source_Capabilities</i> Message and checks the CRC to verify the Message.
16		Physical Layer removes the CRC and forwards the <i>Source_Capabilities</i> Message to the Protocol Layer.
17		Protocol Layer stores the <i>MessageID</i> of the incoming Message. The Protocol Layer forwards the received <i>Source_Capabilities</i> Message information to the Policy Engine that consumes it.
18		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
19	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
20	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
21	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Capabilities</i> Message was successfully sent. Policy Engine stops the <i>SourceCapabilityTimer</i> , stops the <i>NoResponseTimer</i> and starts the <i>SenderResponseTimer</i> .	
USB Power Delivery communication is re-established.		

8.3.2.8 Power Role Swap

8.3.2.8.1 Source Initiated Power Role Swap

8.3.2.8.1.1 Source Initiated Power Role Swap (Accept)

This is an example of a successful Power Role Swap operation initiated by a Port which initially, at the start of this Message sequence, is acting as a Source and therefore has R_p pulled up on its CC wire. It does not include any subsequent Power Negotiation which is required in order to establish an Explicit Contract (see [Section 8.3.2.2 “Power Negotiation”](#)).

There are four distinct phases to the Power Role Swap negotiation:

- A ***PR_Swap*** Message is sent.
- An ***Accept*** Message in response to the ***PR_Swap*** Message.
- The new Sink sets its power output to ***vSafe0V***, then asserts R_d and sends a ***PS_RDY*** Message when this process is complete.
- The new Source asserts R_p , then sets its power output to ***vSafe5V*** and sends a ***PS_RDY*** Message when it is ready to supply power.

Figure 8-35 “Successful Power Role Swap Sequence Initiated by the Source” shows the Messages as they flow across the bus and within the devices to accomplish the Power Role Swap sequence.

Figure 8-35 “Successful Power Role Swap Sequence Initiated by the Source”

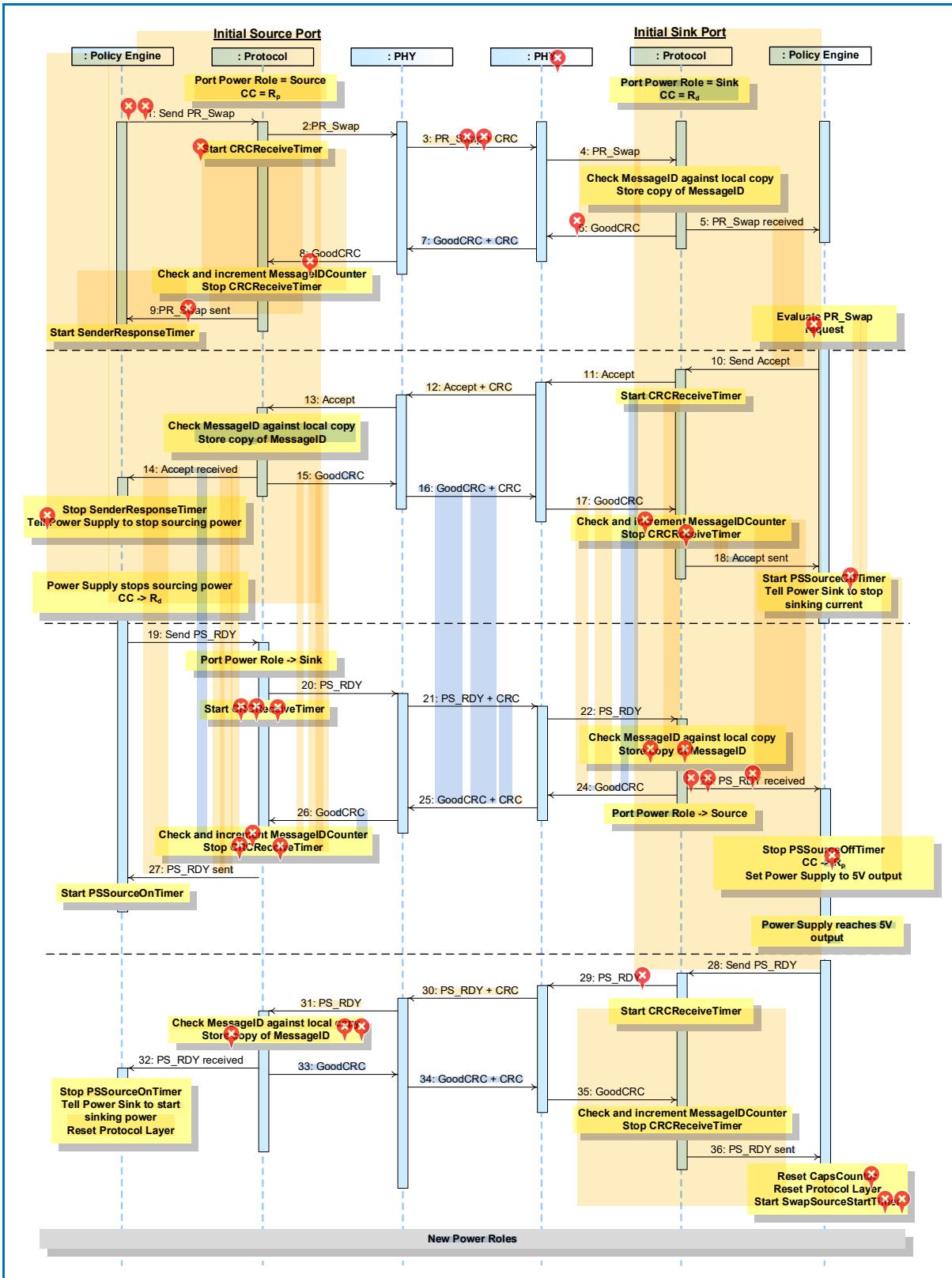


Table 8.63 “Steps for a Successful Source Initiated Power Role Swap Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-35 “Successful Power Role Swap Sequence Initiated by the Source”** above.

Table 8.63 “Steps for a Successful Source Initiated Power Role Swap Sequence”

Step	Initial Source Port	Initially Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a PR_Swap Message.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the PR_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the PR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the PR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received PR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the PR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the PR_Swap Message sent by the Source and decides that it is able and willing to do the Power Role Swap. It tells the Protocol Layer to form an Accept Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Accept Message.	Physical Layer appends a CRC and sends the Accept Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it.	

14	The Policy Engine requests its power supply to stop supplying power and stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine starts the <i>PSSourceOffTimer</i> and tells the power supply to stop sinking current.
19	The Policy Engine determines its power supply is no longer supplying V _{BUS} . The Policy Engine requests the Device Policy Manager to assert the R _d pull down on the CC wire. The Policy Engine then directs the Protocol Layer to generate a <i>PS_RDY</i> Message, with the <i>Port Power Role</i> bit in the Message Header set to "Sink", to tell its Port Partner that it can begin to Source V _{BUS} .	
20	Protocol Layer sets the <i>Port Power Role</i> bit in the Message Header set to "Sink", creates the Message and passes to Physical Layer.	
21	Physical Layer appends CRC and sends the <i>PS_RDY</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the <i>PS_RDY</i> Message and checks the CRC to verify the Message.
22		Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.
23		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>PSSourceOffTimer</i> , directs the Device Policy Manager to apply the R _p pull up and then starts switching the power supply to <i>vSafe5V</i> Source operation. *
24		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
25	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
26	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
27	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. Policy Engine starts <i>PSSourceOnTimer</i> .	

28		Policy Engine, when its power supply is ready to supply power, tells the Protocol Layer to form a <i>PS_RDY</i> Message. The <i>Port Power Role</i> bit used in this, and subsequent Message Headers is now set to “Source”.
29		Protocol Layer creates the <i>PS_RDY</i> Message and passes to Physical Layer.
30	Physical Layer receives the <i>PS_RDY</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>PS_RDY</i> Message. Starts <i>CRCReceiveTimer</i> .
31	Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.	
32	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it.	
33	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
34	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message. The Policy Engine stops the <i>PSSourceOnTimer</i> , informs the power supply it can now Sink power and resets the Protocol Layer.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
35		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
36		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. The Policy Engine resets the <i>CapsCounter</i> , resets the Protocol Layer and starts the <i>SwapSourceStartTimer</i> which must timeout before sending any <i>Source_Capabilities</i> Messages.
The Power Role Swap is complete, the roles have been reversed and the Port Partners are free to negotiate for more power.		

8.3.2.8.1.2

Source Initiated Power Role Swap (Reject)

This is an example of a rejected Power Role Swap operation initiated by a Port which initially, at the start of this Message sequence, is acting as a Source and therefore has R_p pulled up on its CC wire.

There are several phases to the Power Role Swap negotiation:

- A ***PR_Swap*** Message is sent.
- An ***Reject*** Message in response to the ***PR_Swap*** Message.

Figure 8-36 “Rejected Power Role Swap Sequence Initiated by the Source” shows the Messages as they flow across the bus and within the devices.

Figure 8-36 “Rejected Power Role Swap Sequence Initiated by the Source”

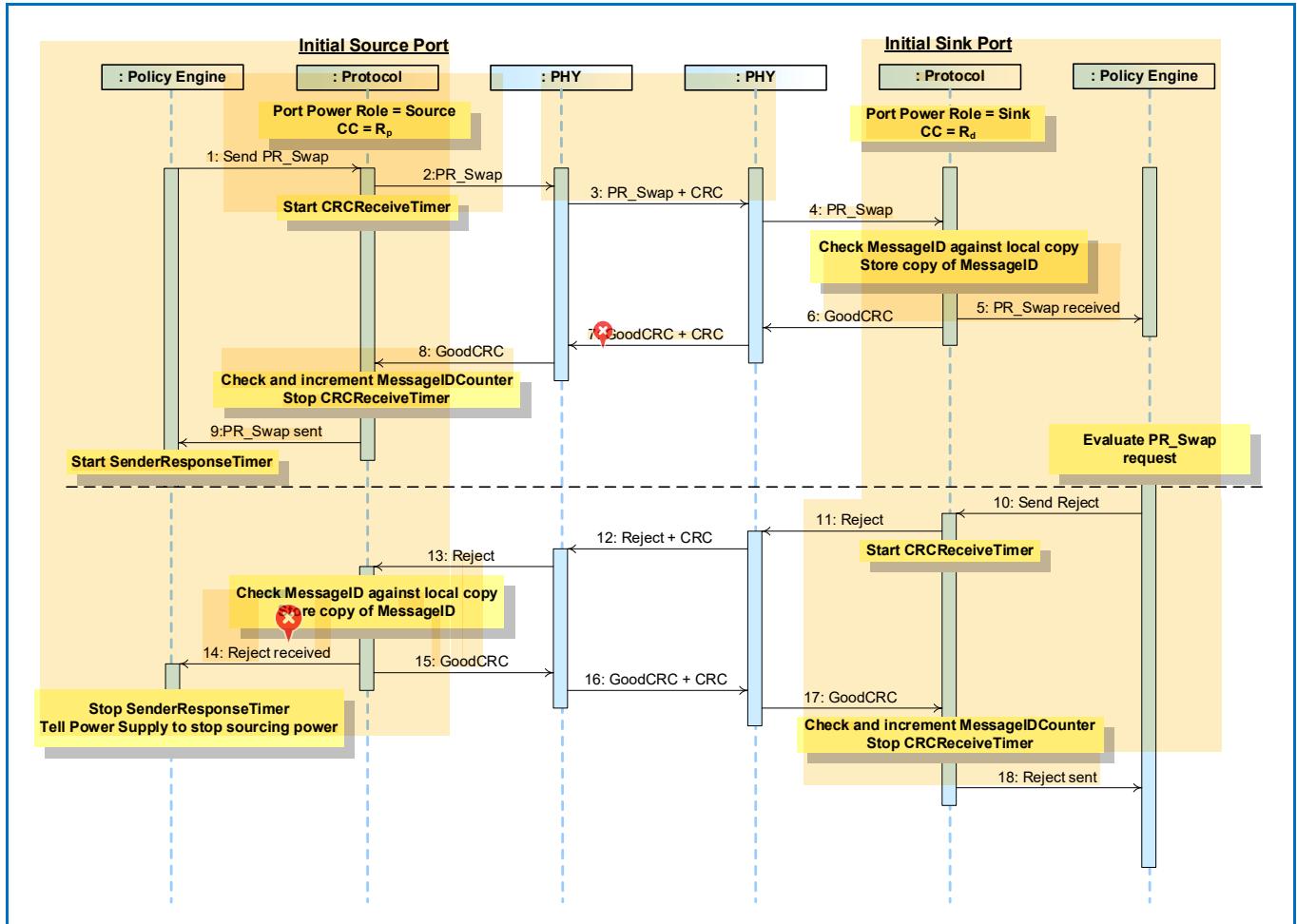


Table 8.64 “Steps for a Rejected Source Initiated Power Role Swap Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-36 “Rejected Power Role Swap Sequence Initiated by the Source”** above.

Table 8.64 “Steps for a Rejected Source Initiated Power Role Swap Sequence”

Step	Initial Source Port	Initially Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a PR_Swap Message.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the PR_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the PR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the PR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received PR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the PR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the PR_Swap Message sent by the Source and decides that it is able and willing to do the Power Role Swap. It tells the Protocol Layer to form a Reject Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Reject Message.	Physical Layer appends a CRC and sends the Reject Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Reject Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Reject</i> Message was successfully sent.

8.3.2.8.1.3

Source Initiated Power Role Swap (Wait)

This is an example of a Power Role Swap operation, with a wait response, initiated by a Port which initially, at the start of this Message sequence, is acting as a Source and therefore has R_p pulled up on its CC wire.

There are several phases to the Power Role Swap negotiation:

- A ***PR_Swap*** Message is sent.
- A ***Wait*** Message in response to the ***PR_Swap*** Message.

Figure 8-37 “Power Role Swap Sequence with wait Initiated by the Source” shows the Messages as they flow across the bus and within the devices.

Figure 8-37 “Power Role Swap Sequence with wait Initiated by the Source”

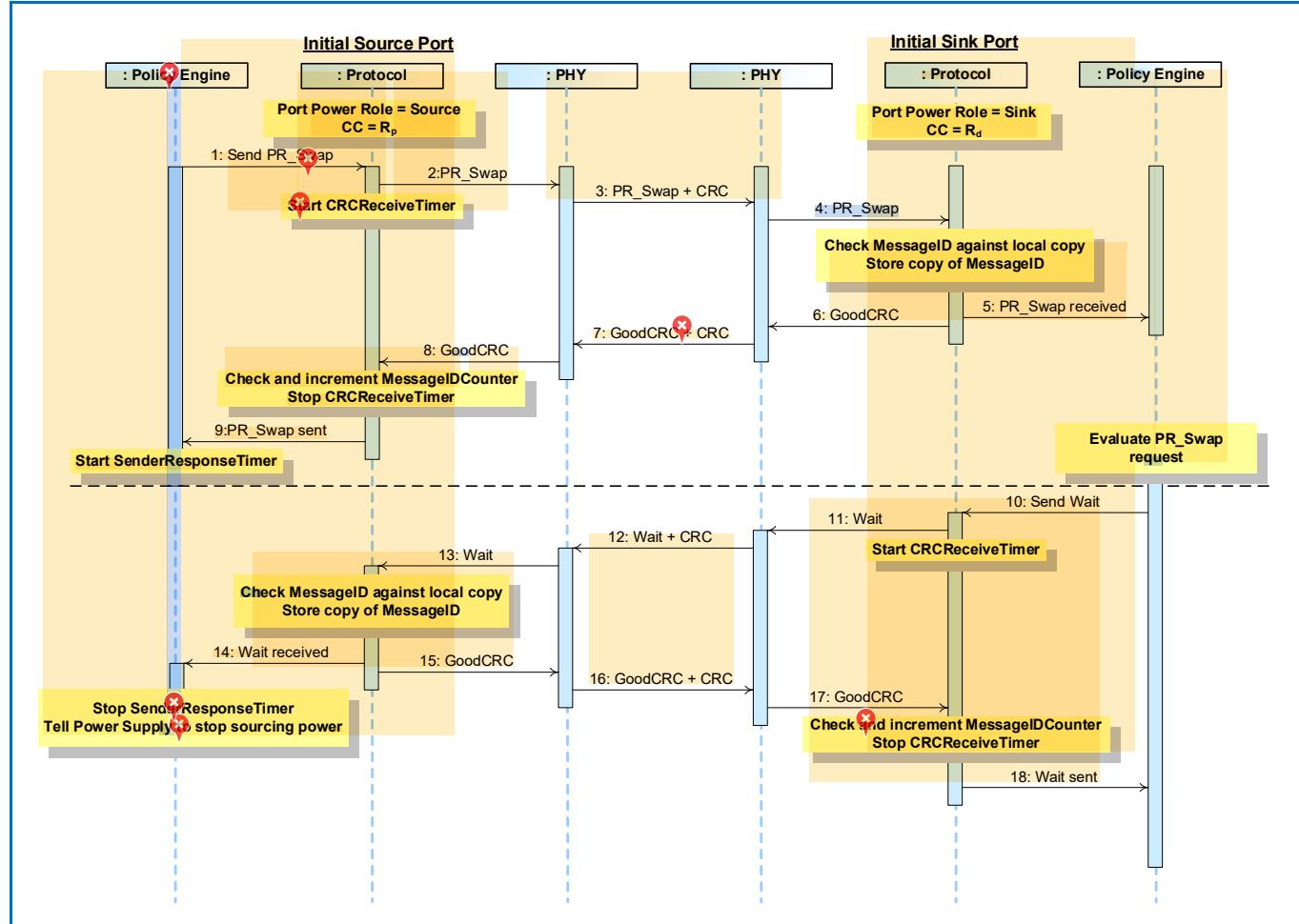


Table 8.65 “Steps for a Source Initiated Power Role Swap with Wait Sequence below provides a detailed explanation of what happens at each labeled step in **Figure 8-37 “Power Role Swap Sequence with wait Initiated by the Source”** above.

Table 8.65 “Steps for a Source Initiated Power Role Swap with Wait Sequence”

Step	Initial Source Port	Initially Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a PR_Swap Message.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the PR_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the PR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the PR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received PR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the PR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the PR_Swap Message sent by the Source and decides that it is able and willing to do the Power Role Swap. It tells the Protocol Layer to form a Wait Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Wait Message.	Physical Layer appends a CRC and sends the Wait Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Wait Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Wait</i> Message was successfully sent.

8.3.2.8.2 Sink Initiated Power Role Swap

8.3.2.8.2.1 Sink Initiated Power Role Swap (Accept)

This is an example of a successful Power Role Swap operation initiated by a Port which initially, at the start of this Message sequence, is acting as a Sink and therefore has R_d pulled down on its CC wire. It does not include any subsequent Power Negotiation which is required in order to establish an Explicit Contract (see [Section 8.3.2.2 "Power Negotiation"](#)).

There are four distinct phases to the Power Role Swap negotiation:

- A ***PR_Swap*** Message is sent.
- An ***Accept*** Message in response to the ***PR_Swap*** Message.
- The new Sink sets its power output to ***vSafe0V***, then asserts R_d and sends a ***PS_RDY*** Message when this process is complete.
- The new Source asserts R_p , then sets its power output to ***vSafe5V*** and sends a ***PS_RDY*** Message when it is ready to supply power.

Figure 8-38 “Successful Power Role Swap Sequence Initiated by the Sink” shows the Messages as they flow across the bus and within the devices to accomplish the Power Role Swap.

Figure 8-38 “Successful Power Role Swap Sequence Initiated by the Sink”

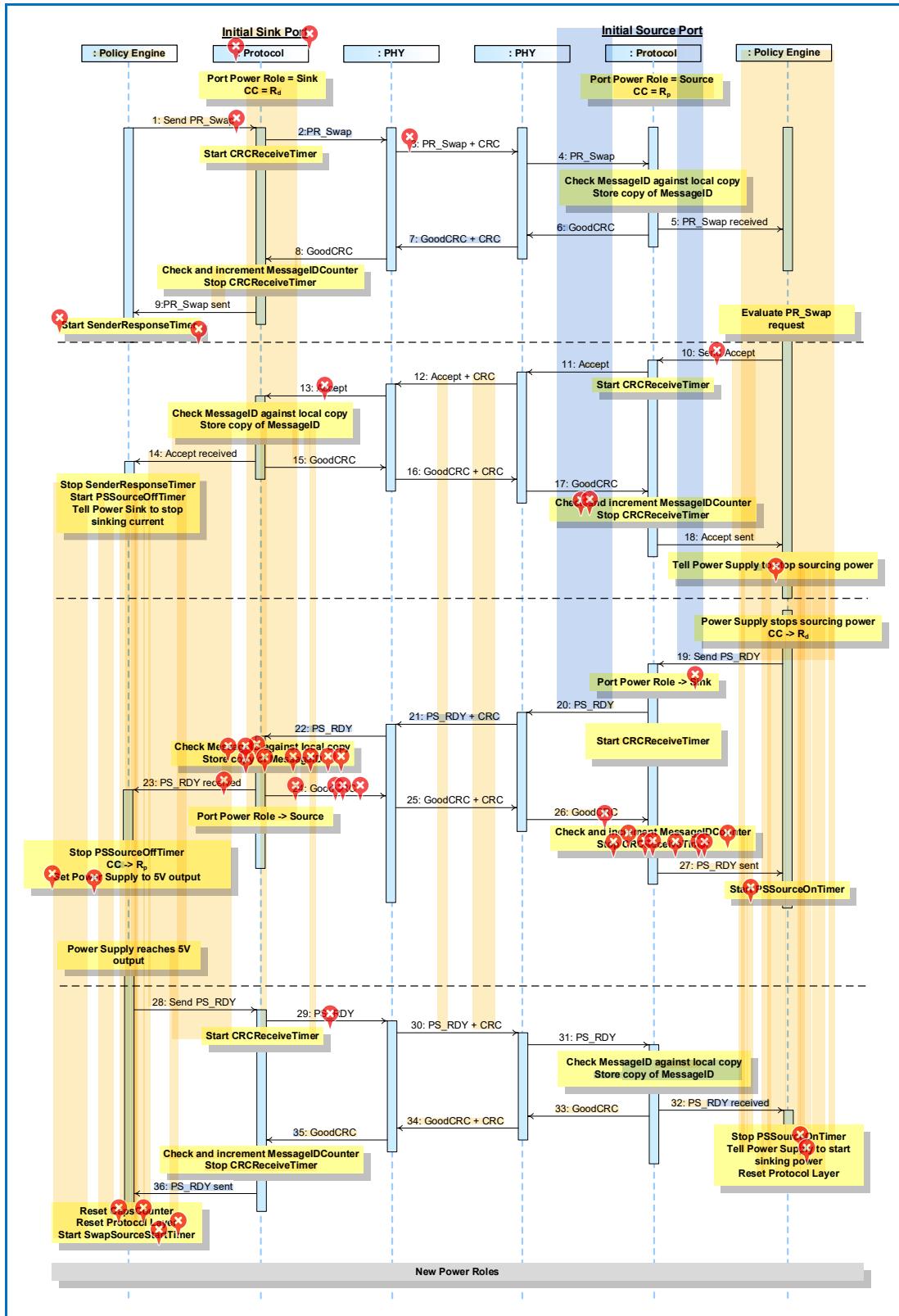


Table 8.66 “Steps for a Successful Sink Initiated Power Role Swap Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-38 “Successful Power Role Swap Sequence Initiated by the Sink”** above.

✖**Table 8.66 “Steps for a Successful Sink Initiated Power Role Swap Sequence”**

Step	Initial Sink Port	Initial Source Port
1	The Port has Port Power Role set to Sink with the R _d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a PR_Swap Message.	The Port has Port Power Role set to Source and the R _p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the PR_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the PR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the PR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received PR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the PR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the PR_Swap Message sent by the Sink and decides that it is able and willing to do the Power Role Swap. It tells the Protocol Layer to form an Accept Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Accept Message.	Physical Layer appends a CRC and sends the Accept Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>SenderResponseTimer</i> , starts the <i>PSSourceOffTimer</i> and tells the power supply to stop sinking current.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine tells the power supply to stop supplying power.
19		The Policy Engine determines its power supply is no longer supplying V_{BUS} . The Policy Engine requests the Device Policy Manager to assert the R_d pull down on the CC wire. The Policy Engine then directs the Protocol Layer to generate a <i>PS_RDY</i> Message, with the <i>Port Power Role</i> bit in the Message Header set to "Sink", to tell its Port Partner that it can begin to Source V_{BUS} .
20		Protocol Layer sets the <i>Port Power Role</i> bit in the Message Header set to "Sink", creates the Message and passes to Physical Layer.
21	Physical Layer receives the <i>PS_RDY</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>PS_RDY</i> Message. Starts <i>CRCReceiveTimer</i> .
22	Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.	
23	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>PSSourceOffTimer</i> , directs the Device Policy Manager to apply the R_p pull up and then starts switching the power supply to <i>vSafe5V</i> Source operation.	
24	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
25	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.
26		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
27		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. Policy Engine starts <i>PSSourceOnTimer</i> .

28	Policy Engine, when its power supply is ready to supply power, tells the Protocol Layer to form a <i>PS_RDY</i> Message. The <i>Port Power Role</i> bit used in this, and subsequent Message Headers is now set to "Source".	
29 X	Protocol Layer creates the <i>PS_RDY</i> Message and passes to Physical Layer.	
30	Physical Layer appends a CRC and sends the <i>PS_RDY</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the <i>PS_RDY</i> Message and compares the CRC it calculated with the one sent to verify the Message.
31		Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.
32 X		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>PSSourceOnTimer</i> , informs the power supply that it can start consuming power.
33		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
34	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message. The Policy Engine stops the <i>PSSourceOnTimer</i> , informs the power supply it can now Sink power and resets the Protocol Layer.
35	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> to the Protocol Layer.	
36	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent. The Policy Engine resets the <i>CapsCounter</i> , resets the Protocol Layer and starts the <i>SwapSourceStartTimer</i> which must timeout before sending any <i>Source_Capabilities</i> Messages.	
The Power Role Swap is complete, the roles have been reversed and the Port Partners are free to negotiate for more power.		

8.3.2.8.2.2

Sink Initiated Power Role Swap (Reject)

This is an example of a rejected Power Role Swap operation initiated by a Port which initially, at the start of this Message sequence, is acting as a Sink and therefore has R_d pulled down on its CC wire.

There are several phases to the Power Role Swap negotiation:

- A *PR_Swap* Message is sent.
- A *Reject* Message in response to the *PR_Swap* Message.

Figure 8-39 “Rejected Power Role Swap Sequence Initiated by the Sink” shows the Messages as they flow across the bus and within the devices.

Figure 8-39 “Rejected Power Role Swap Sequence Initiated by the Sink”

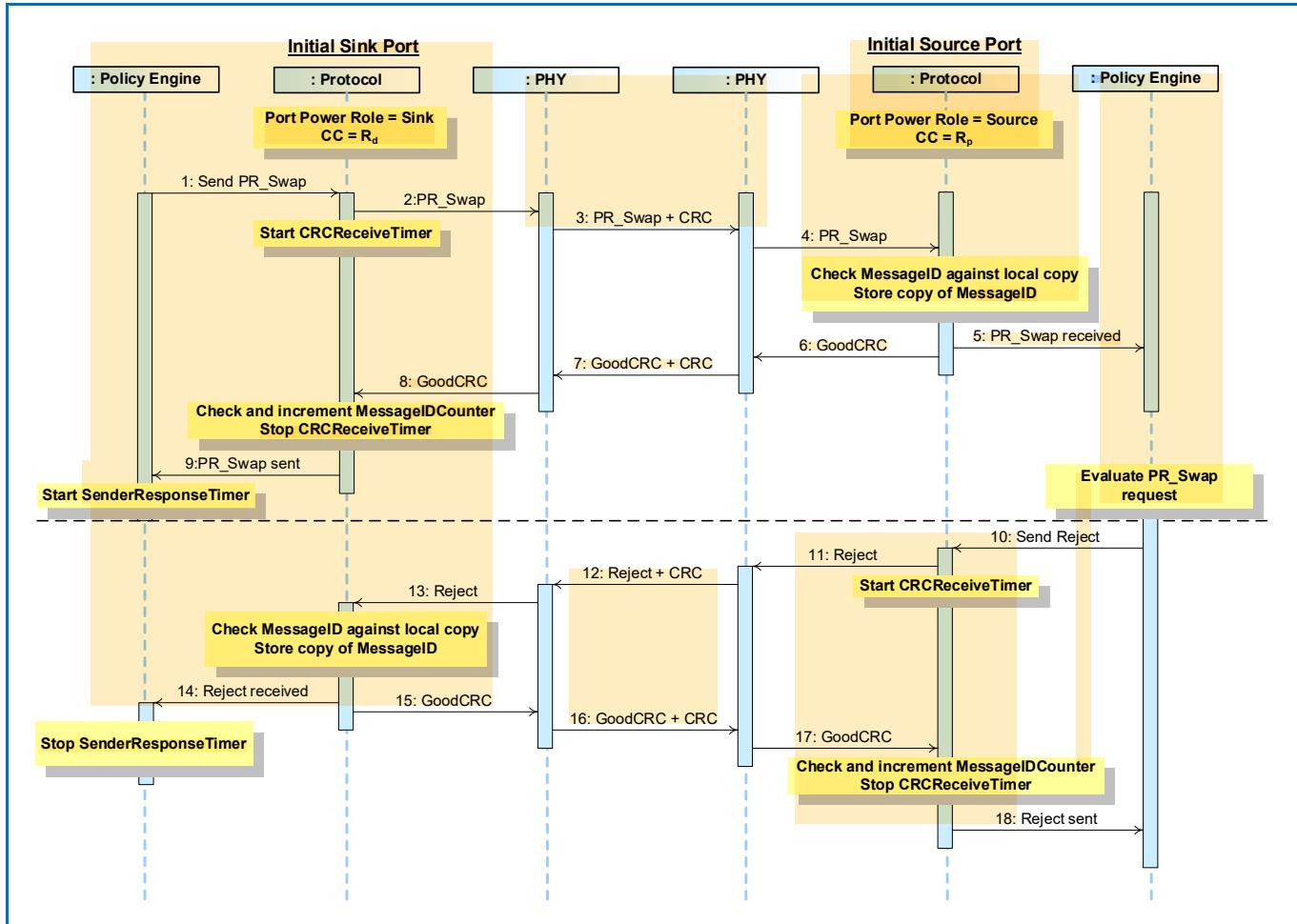


Table 8.67 “Steps for a Rejected Sink Initiated Power Role Swap Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-39 “Rejected Power Role Swap Sequence Initiated by the Sink”** above.

Table 8.67 “Steps for a Rejected Sink Initiated Power Role Swap Sequence”

Step	Initial Sink Port	Initial Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a PR_Swap Message.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the PR_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the PR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the PR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received PR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the PR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the PR_Swap Message sent by the Sink and decides that it is able and willing to do the Power Role Swap. It tells the Protocol Layer to form a Reject Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Reject Message.	Physical Layer appends a CRC and sends the Accept Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Reject Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and  passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the  <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Reject</i> Message was successfully sent

8.3.2.8.2.3

Sink Initiated Power Role Swap (Wait)

This is an example of a Power Role Swap operation, responded to with wait, initiated by a Port which initially, at the start of this Message sequence, is acting as a Sink and therefore has R_d pulled down on its CC wire.

There are several phases to the Power Role Swap negotiation:

- A ***PR_Swap*** Message is sent.
- A ***Wait*** Message in response to the ***PR_Swap*** Message.

Figure 8-40 “Power Role Swap Sequence with wait Initiated by the Sink” shows the Messages as they flow across the bus and within the devices.

Figure 8-40 “Power Role Swap Sequence with wait Initiated by the Sink”

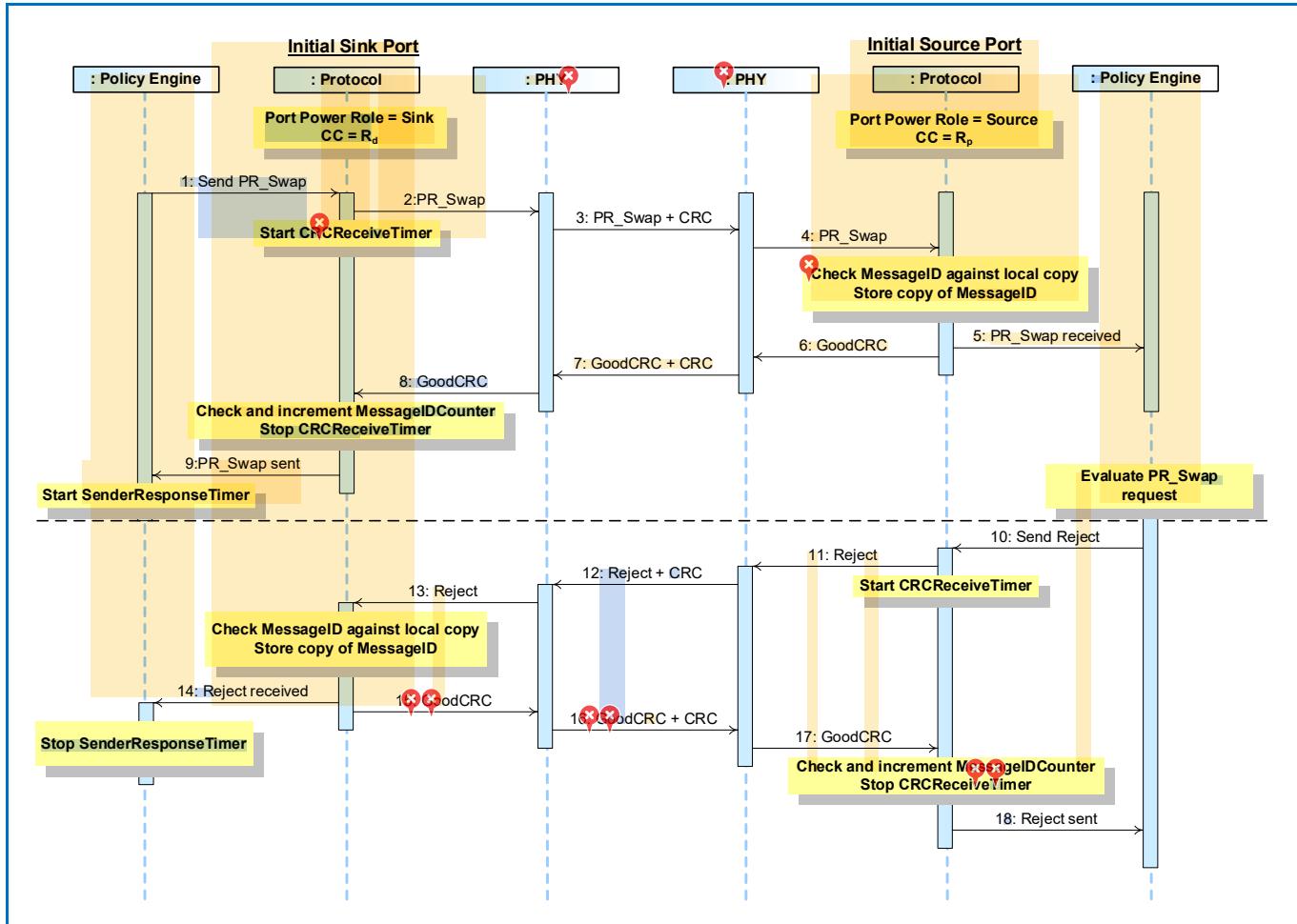


Table 8.68 “Steps for a Sink Initiated Power Role Swap with Wait Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-40 “Power Role Swap Sequence with wait Initiated by the Sink”** above.

Table 8.68 “Steps for a Sink Initiated Power Role Swap with Wait Sequence”

Step	Initial Sink Port	Initial Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a PR_Swap Message.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the PR_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the PR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the PR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received PR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the PR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the PR_Swap Message sent by the Sink and decides that it is able and willing to do the Power Role Swap. It tells the Protocol Layer to form a Wait Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Wait Message.	Physical Layer appends a CRC and sends the Wait Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Wait Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the <i>Wait</i> Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Wait</i> Message was successfully sent

8.3.2.9 Fast Role Swap

This is an example of a successful Fast Role Swap operation initiated by a Port that is initially a Source and therefore has R_p pulled up on its CC Wire and which has lost power and needs to get $vSafe5V$ quickly. It does not include any subsequent Power Negotiation which is required in order to establish an Explicit Contract (see [Section 8.3.2.2 "Power Negotiation"](#)).

There are several distinct phases to the Fast Role Swap negotiation:

- The initial Source stops driving its power output which starts transitioning to $vSafe0V$ and signals Fast Role Swap on the CC Wire; these could occur in either order or simultaneously.
- The initial Sink stops Sinking power. At this point the new Source still has R_d asserted and the new Sink still has R_p asserted.
- An FR_Swap Message is sent by the new Source within $tFRSwapInit$ of detecting the Fast Swap signal.
- An $Accept$ Message is sent by the new Sink in response to the FR_Swap Message.
- The new Sink asserts R_d and sends a PS_RDY Message indicating that the Voltage on V_{BUS} is at or below $vSafe5V$.
- The new Source asserts R_p and sends a PS_RDY Message indicating that it is acting as a Source and is supplying $vSafe5V$.

Note: that the new Source can start applying V_{BUS} when V_{BUS} is at or below $vSafe5V$ (max) but will start driving V_{BUS} to $vSafe5V$ no later than $tSrcFRSwap$ after detecting both the FRS signal and that V_{BUS} has dropped below $vSafe5V$ (min).

[Figure 8-41 "Successful Fast Role Swap Sequence"](#) shows the Messages as they flow across the bus and within the devices to accomplish the Fast Role Swap.

Figure 8-41 “Successful Fast Role Swap Sequence”

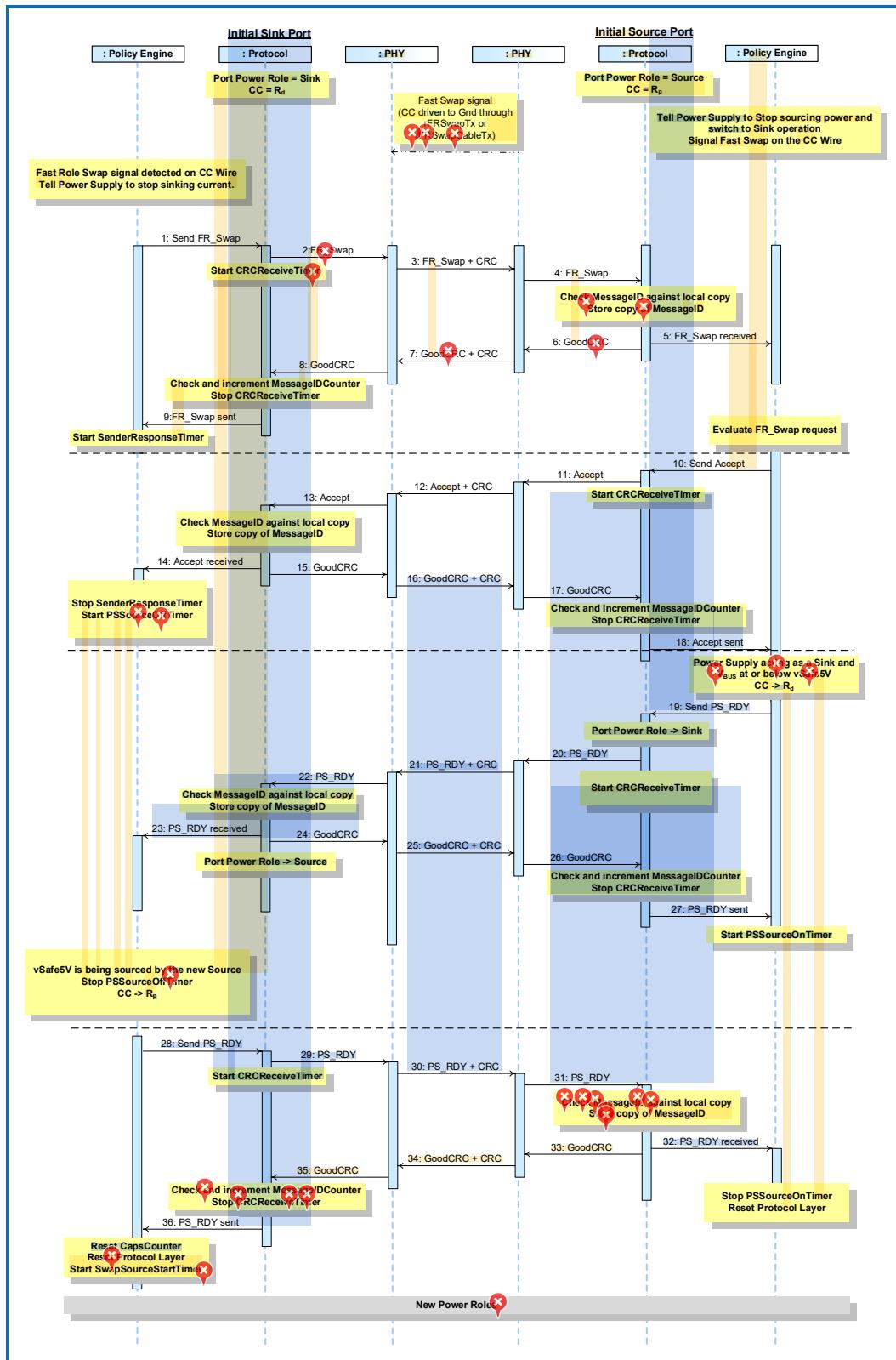


Table 8.69 “Steps for a Successful Fast Role Swap Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-41 “Successful Fast Role Swap Sequence”** above.

Table 8.69 “Steps for a Successful Fast Role Swap Sequence”

Step	Initial Sink Port	Initial Source Port
1	<p>The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.</p> <p>The Device Policy Manager detects Fast Swap on the CC Wire and tells the power supply to stop sinking current.</p> <p>The Policy Engine directs the Protocol Layer to send an FR_Swap Message within tFRSwapInit of detecting the Fast Swap signal.</p>	<p>The Port has Port Power Role set to Source and the R_p pull up on its CC wire.</p> <p>The Device Policy Manager tells the Power Supply to stop sourcing power and switch to Sink operation.</p> <p>The Device Policy Manager signals Fast Swap on the CC Wire by driving CC to ground with a resistance of less than rFRSwapTx for at least tFRSwapTx.</p>
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the FR_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the FR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the PR_Swap Message to the Protocol Layer.
5		<p>Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value.</p> <p>The Protocol Layer forwards the received FR_Swap Message information to the Policy Engine that consumes it.</p>
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	<p>Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer.</p> <p>Protocol Layer informs the Policy Engine that the FR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer.</p>	
10		Policy Engine evaluates the PR_Swap Message sent by the Sink and decides that it is able and willing to do the Power Role Swap. It tells the Protocol Layer to form an Accept Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Accept Message.	Physical Layer appends a CRC and sends the Accept Message. Starts CRCReceiveTimer .

13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PR_Swap</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>SenderResponseTimer</i> , starts the <i>PSSourceOffTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent.
19		The Policy Engine determines its power supply is no longer supplying V _{BUS} and is acting as a Sink. The Policy Engine requests the Device Policy Manager to assert the R _d pull down on the CC wire. The Policy Engine then directs the Protocol Layer to generate a <i>PS_RDY</i> Message, with the <i>Port Power Role</i> bit in the Message Header set to "Sink", to tell its Port Partner that it can begin to Source V _{BUS} .
20		Protocol Layer sets the <i>Port Power Role</i> bit in the Message Header set to "Sink", creates the Message and passes to Physical Layer.
21	Physical Layer receives the <i>PS_RDY</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>PS_RDY</i> Message. Starts <i>CRCReceiveTimer</i> .
22	Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.	
23	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it. The Policy Engine stops the <i>PSSourceOffTimer</i> .	
24	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
25	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.
26		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.

27		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the PS_RDY Message was successfully sent. Policy Engine starts PSSourceOnTimer .
28	<p>The Policy Engine directs the Device Policy Manager to apply the R_p pull up.</p> <p>Note: at some point (either before or after receiving the PS_RDY Message) the new Source has applied vSafe5V no later than tSrcFRSwap after detecting the FRS signal and that V_{BUS} has dropped below vSafe5V.</p> <p>Policy Engine, when its power supply is ready to supply power, tells the Protocol Layer to form a PS_RDY Message. The Port Power Role bit used in this, and subsequent Message Headers is now set to "Source".</p>	
29	Protocol Layer creates the PS_RDY Message and passes to Physical Layer.	
30	Physical Layer appends a CRC and sends the PS_RDY Message. Starts CRCReceiveTimer .	Physical Layer receives the PS_RDY Message and compares the CRC it calculated with the one sent to verify the Message.
31		Physical Layer removes the CRC and forwards the PS_RDY Message to the Protocol Layer.
32		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received PS_RDY Message information to the Policy Engine that consumes it. The Policy Engine stops the PSSourceOnTimer .
33		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
34	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the GoodCRC Message. The Policy Engine resets the Protocol Layer.
35	Physical Layer removes the CRC and forwards the GoodCRC to the Protocol Layer.	
36	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the PS_RDY Message was successfully sent. The Policy Engine resets the CapsCounter , resets the Protocol Layer and starts the SwapSourceStartTimer which must timeout before sending any Source_Capabilities Messages.	

The Fast Role Swap is complete, the roles have been reversed and the Port Partners are free to negotiate for more power.

8.3.2.10 Data Role Swap

8.3.2.10.1 Data Role Swap, Initiated by UFP Operating as Sink

8.3.2.10.1.1 Data Role Swap, Initiated by UFP Operating as Sink (Accept)

Figure 8-42 “Data Role Swap, UFP operating as Sink initiates” shows an example sequence between a Port, which is initially a UFP (Device) and a Sink (R_d asserted), and a Port which is initially a DFP (Host) and a Source (R_p asserted). A Data Role Swap is initiated by the UFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and R_p/R_d remain constant) but exchange data roles between DFP (Host) and UFP (Device).

Figure 8-42 “Data Role Swap, UFP operating as Sink initiates”

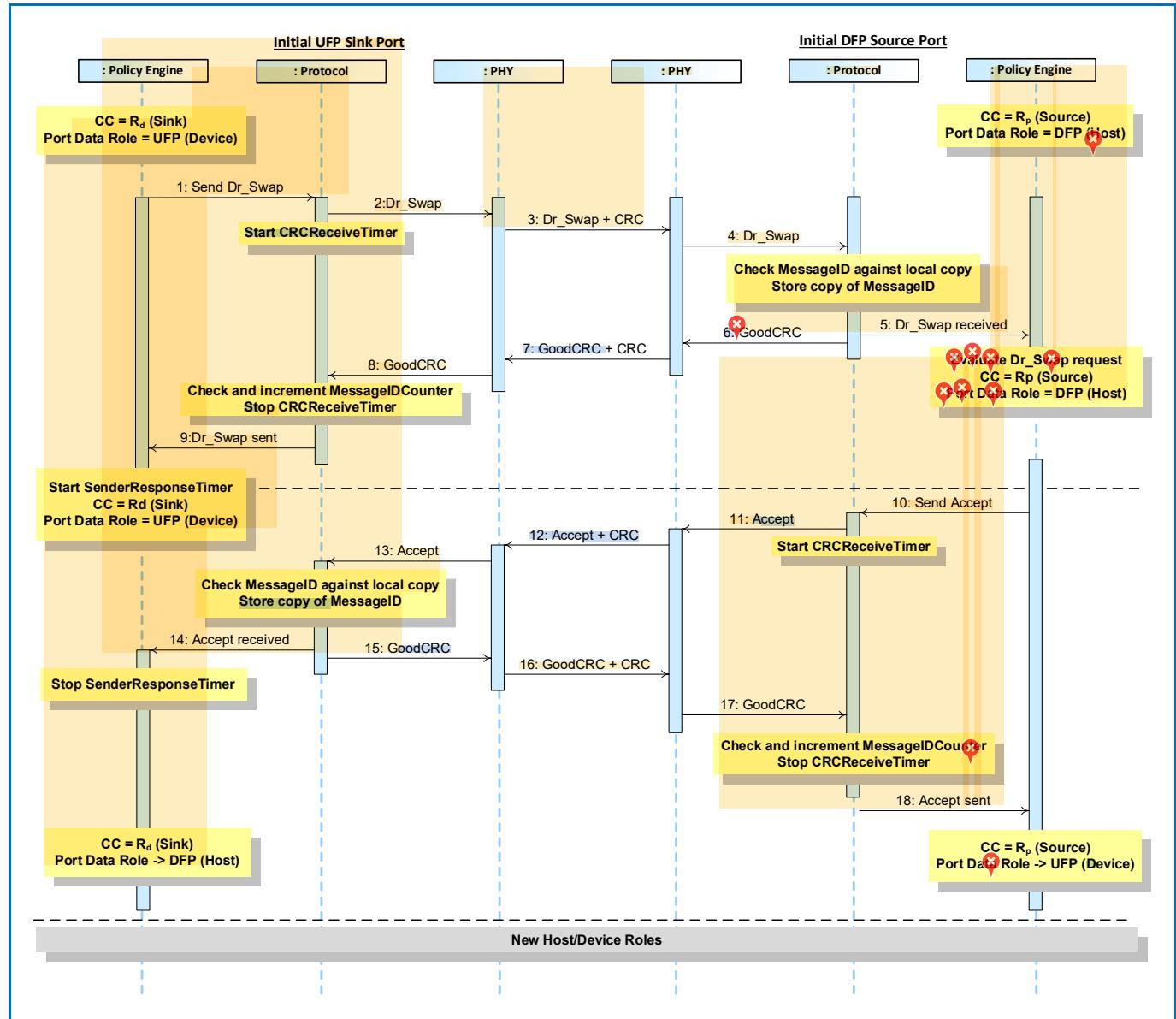


Table 8.70 “Steps for Data Role Swap, UFP operating as Sink initiates” below provides a detailed explanation of what happens at each labeled step in **Figure 8-42 “Data Role Swap, UFP operating as Sink initiates”** above.

📍 **Table 8.70 “Steps for Data Role Swap, UFP operating as Sink initiates”**

Step	Initial UFP Sink Port	Initial DFP Source Port
1	Port starts as a UFP (Device) operating as a Sink with R_d asserted and Port Data Role set to UFP. The Policy Engine directs the Protocol Layer to send a DR_Swap Message.	Port starts as a DFP (Host) operating as Source with R_p asserted and Port Data Role set to DFP.
2	Protocol Layer creates the DR_Swap Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the DR_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the DR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the DR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received DR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the DR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the DR_Swap Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an Accept Message.
11		Protocol Layer creates the Accept Message and passes to Physical Layer.
12	Physical Layer receives the Accept Message and checks the CRC to verify the Message.	Physical Layer appends a CRC and sends the Accept Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	

16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18	<p>The Policy Engine requests that Data Role is changed from UFP (Device) to DFP (Host).</p> <p>The Power Delivery role is now a DFP (Host), with <i>Port Data Role</i> set to DFP, still operating as a Sink (R_d asserted).</p>	<p>Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i>.</p> <p>Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent.</p> <p>The Policy Engine requests that the Data Role is changed to UFP (Device), with <i>Port Data Role</i> set to UFP and continues supplying power as a Source (R_p asserted).</p>
The Data Role Swap is complete; the data roles have been reversed while maintaining the direction of power flow.		

8.3.2.10.1.2

Data Role Swap, Initiated by UFP Operating as Sink (Reject)

Figure 8-43 “Rejected Data Role Swap, UFP operating as Sink initiates” shows an example sequence between a Port, which is initially a UFP (Device) and a Sink (R_d asserted), and a Port which is initially a DFP (Host) and a Source (R_p asserted). A Data Role Swap is initiated by the UFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and R_p/R_d remain constant) and the exchange of data roles is rejected.

Figure 8-43 “Rejected Data Role Swap, UFP operating as Sink initiates”

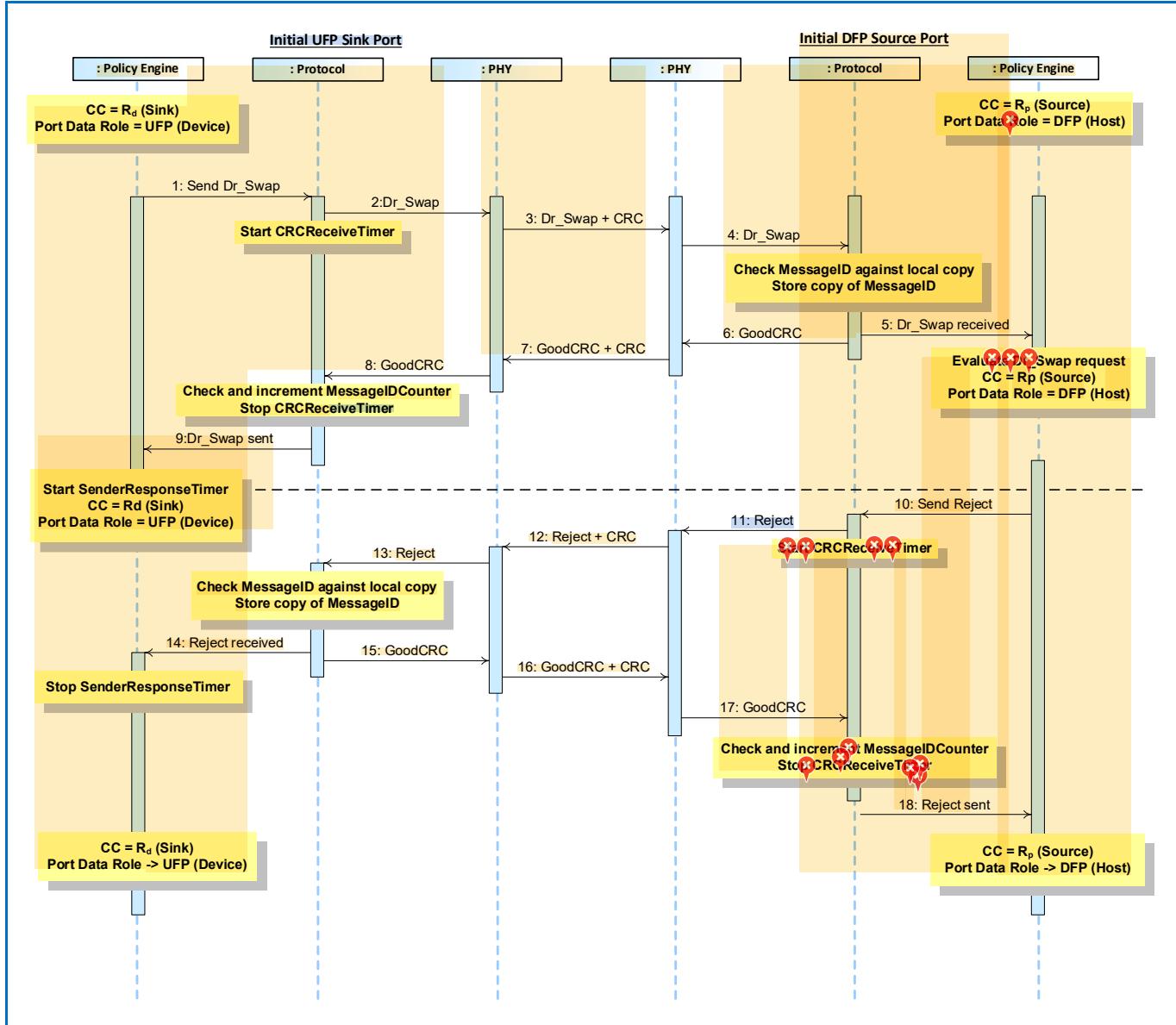


Table 8.71 “Steps for Rejected Data Role Swap, UFP operating as Sink initiates” below provides a detailed explanation of what happens at each labeled step in **Figure 8-43 “Rejected Data Role Swap, UFP operating as Sink initiates”** above.

Table 8.71 “Steps for Rejected Data Role Swap, UFP operating as Sink initiates”

Step	Initial UFP Sink Port	Initial DFP Source Port
1	Port starts as a UFP (Device) operating as a Sink with R_d asserted and Port Data Role set to UFP. The Policy Engine directs the Protocol Layer to send a DR_Swap Message.	Port starts as a DFP (Host) operating as Source with R_p asserted and Port Data Role set to DFP.
2	Protocol Layer creates the DR_Swap Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the DR_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the DR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the DR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received DR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the DR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the DR_Swap Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form a Reject Message.
11		Protocol Layer creates the Reject Message and passes to Physical Layer.
12	Physical Layer receives the Reject Message and checks the CRC to verify the Message.	Physical Layer appends a CRC and sends the Reject Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Reject Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Reject</i> Message was successfully sent.

8.3.2.10.1.3

Data Role Swap, Initiated by UFP Operating as Sink (Wait)

Figure 8-44 “Data Role Swap with Wait, UFP operating as Sink initiates” shows an example sequence between a Port, which is initially a UFP (Device) and a Sink (R_d asserted), and a Port which is initially a DFP (Host) and a Source (R_p asserted). A Data Role Swap is initiated by the UFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and R_p/R_d remain constant) and the exchange of data roles is delayed with a wait.

Figure 8-44 “Data Role Swap with Wait, UFP operating as Sink initiates”

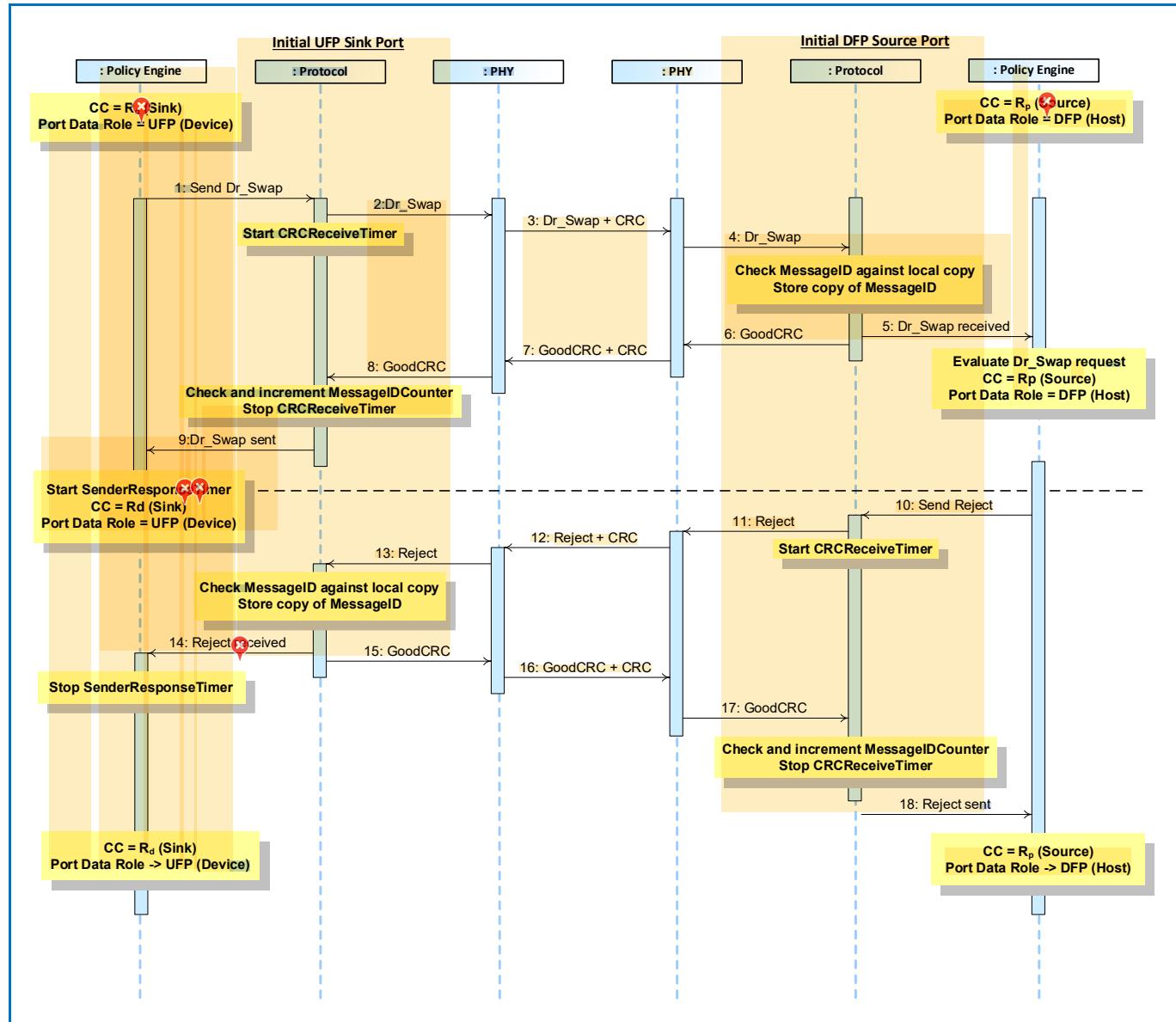


Table 8.72 “Steps for Data Role Swap with Wait, UFP operating as Sink initiates” below provides a detailed explanation of what happens at each labeled step in **Figure 8-44 “Data Role Swap with Wait, UFP operating as Sink initiates”** above.

Table 8.72 “Steps for Data Role Swap with Wait, UFP operating as Sink initiates”

Step	Initial UFP Sink Port	Initial DFP Source Port
1	Port starts as a UFP (Device) operating as a Sink with R_d asserted and Port Data Role set to UFP. The Policy Engine directs the Protocol Layer to send a DR_Swap Message.	Port starts as a DFP (Host) operating as Source with R_p asserted and Port Data Role set to DFP.
2	Protocol Layer creates the DR_Swap Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the DR_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the DR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the DR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received DR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the DR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the DR_Swap Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form a Wait Message.
11		Protocol Layer creates the Wait Message and passes to Physical Layer.
12	Physical Layer receives the Wait Message and checks the CRC to verify the Message.	Physical Layer appends a CRC and sends the Wait Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Wait Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC?</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Wait</i> Message was successfully sent.

8.3.2.10.2 Data Role Swap, Initiated by UFP Operating as Source

8.3.2.10.2.1 Data Role Swap, Initiated by UFP Operating as Source (Accept)

Figure 8-45 “Data Role Swap, UFP operating as Source initiates” shows an example sequence between a Port, which is initially a UFP (Device) and a Source (R_p asserted), and a Port which is initially a DFP (Host) and a Sink (R_d asserted). A Data Role Swap is initiated by the UFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and R_p/R_d remain constant) but exchange data roles between DFP (Host) and UFP (Device).

Figure 8-45 “Data Role Swap, UFP operating as Source initiates”

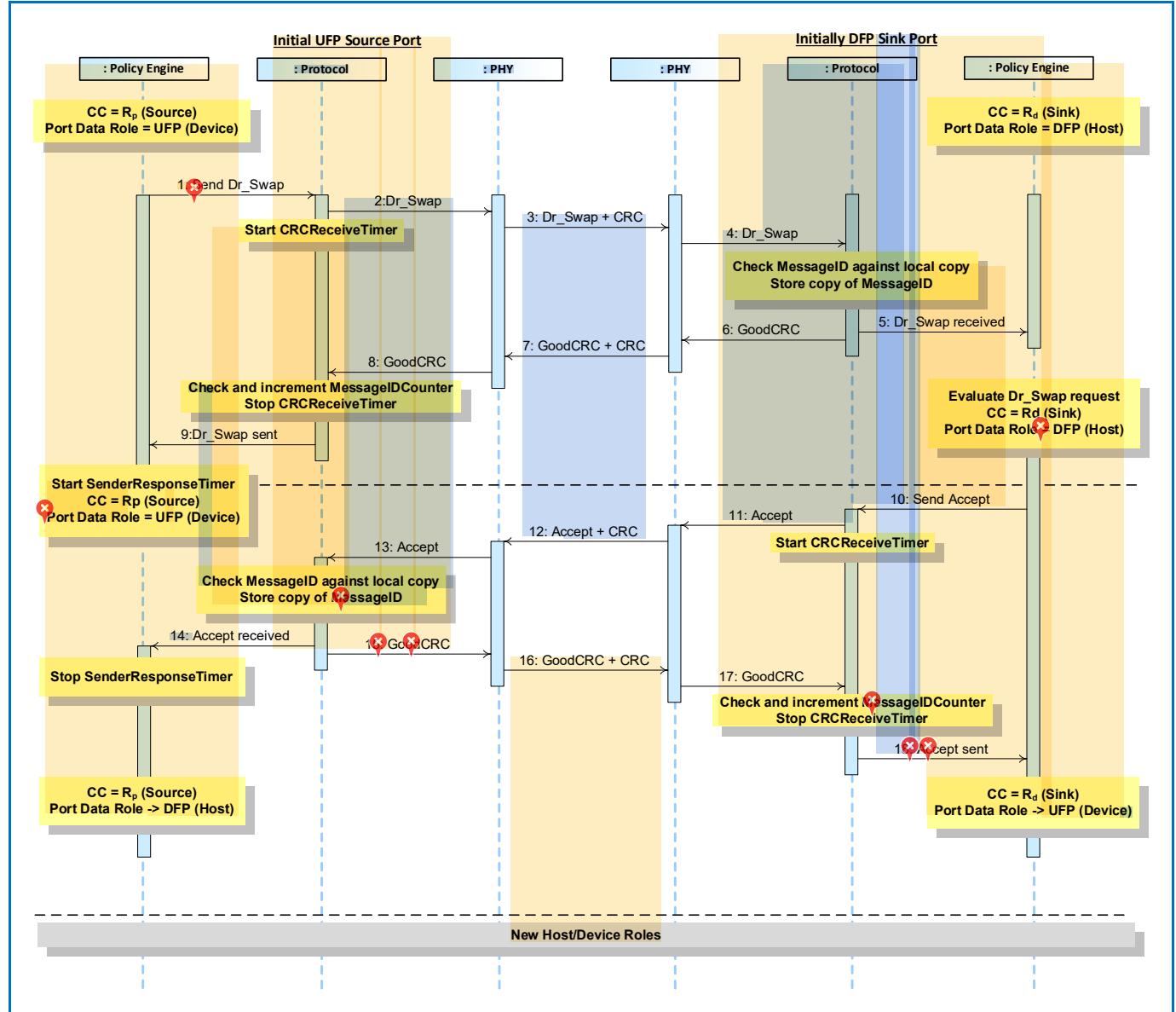


Table 8.73 "Steps for Data Role Swap, UFP operating as Source initiates" below provides a detailed explanation of what happens at each labeled step in **Figure 8-45 "Data Role Swap, UFP operating as Source initiates"** above.

📍 **Table 8.73 "Steps for Data Role Swap, UFP operating as Source initiates"**

Step	Initial UFP Source Port	Initial DFP Sink Port
1	Port starts as a UFP (Device) operating as Source with R_p asserted and Port Data Role set to UFP. The Policy Engine directs the Protocol Layer to send a DR_Swap Message.	Port starts as a DFP (Host) operating as a Sink with R_d asserted and Port Data Role set to DFP.
2	Protocol Layer creates the DR_Swap Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the DR_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the DR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the DR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received DR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the DR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the DR_Swap Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an Accept Message.
11		Protocol Layer creates the Accept Message and passes to Physical Layer.
12	Physical Layer receives the Accept Message and checks the CRC to verify the Message.	Physical Layer appends a CRC and sends the Accept Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	

16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18	<p>The Policy Engine requests that Data Role is changed from UFP (Device) to DFP (Host).</p> <p>The Power Delivery role is now a DFP (Host), and <i>Port Data Role</i> set to DFP and continues supplying power as a Source (R_p asserted).</p>	<p>Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i>.</p> <p>Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine requests that the Data Role is changed to UFP (Device), with <i>Port Data Role</i> set to UFP and still operating as a Sink (R_p asserted).</p>

The Data Role Swap is complete; the data roles have been reversed while maintaining the direction of power flow.

8.3.2.10.2.2

Data Role Swap, Initiated by UFP Operating as Source (Reject)

Figure 8-46 “Rejected Data Role Swap, UFP operating as Source initiates” shows an example sequence between a Port, which is initially a UFP (Device) and a Source (R_p asserted), and a Port which is initially a DFP (Host) and a Sink (R_d asserted). A Data Role Swap is initiated by the UFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and R_p/R_d remain constant) and the exchange of data roles is rejected.

Figure 8-46 “Rejected Data Role Swap, UFP operating as Source initiates”

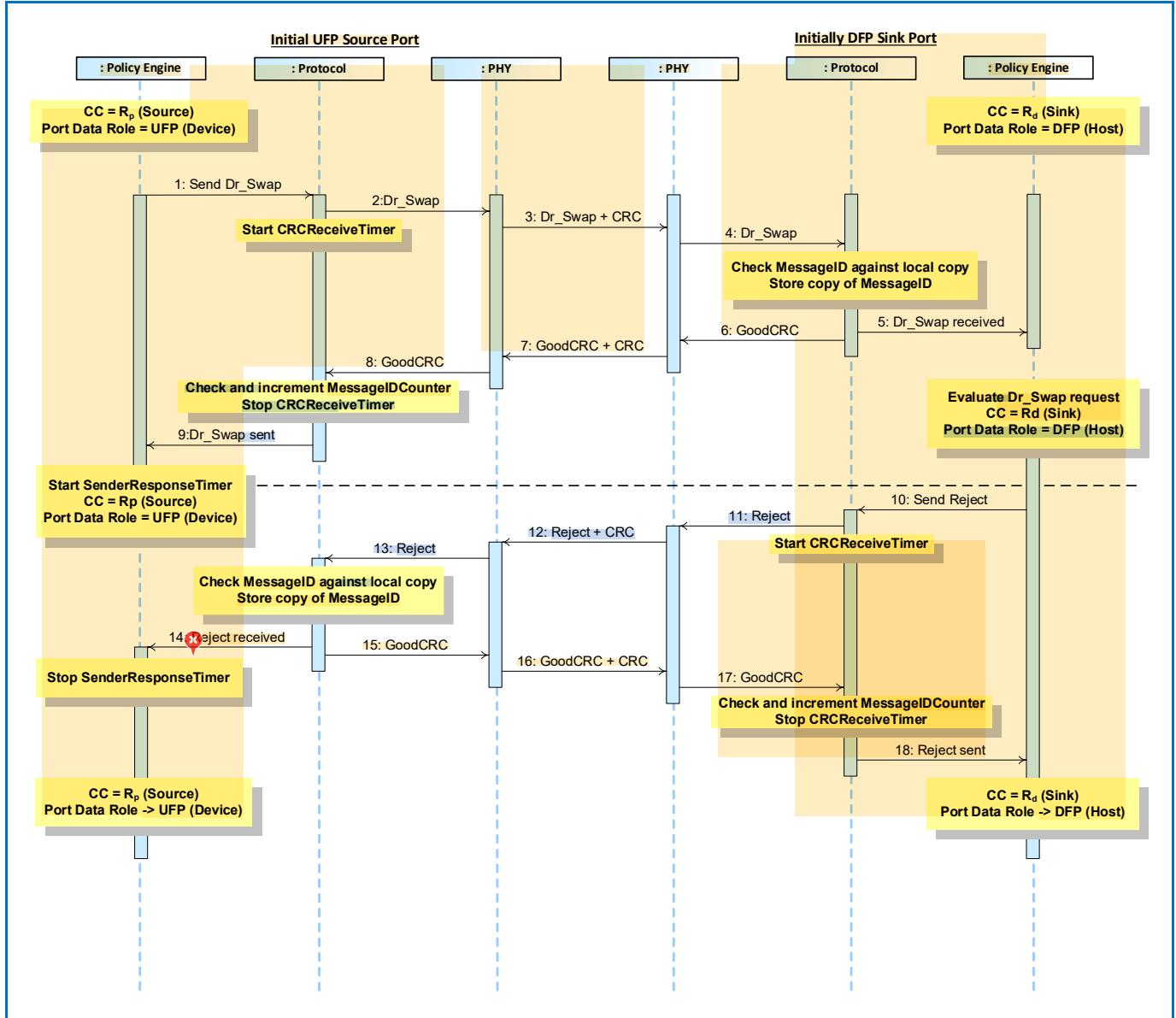


Table 8.74 “Steps for Rejected Data Role Swap, UFP operating as Source initiates” below provides a detailed explanation of what happens at each labeled step in **Figure 8-46 “Rejected Data Role Swap, UFP operating as Source initiates”** above.

Table 8.74 “Steps for Rejected Data Role Swap, UFP operating as Source initiates”

Step	Initial UFP Source Port	Initial DFP Sink Port
1	Port starts as a UFP (Device) operating as Source with R_p asserted and Port Data Role set to UFP. The Policy Engine directs the Protocol Layer to send a DR_Swap Message.	Port starts as a DFP (Host) operating as a Sink with R_d asserted and Port Data Role set to DFP.
2	Protocol Layer creates the DR_Swap Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the DR_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the DR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the DR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received DR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the DR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the DR_Swap Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form a Reject Message.
11		Protocol Layer creates the Reject Message and passes to Physical Layer.
12	Physical Layer receives the Reject Message and checks the CRC to verify the Message.	Physical Layer appends a CRC and sends the Reject Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Reject Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Reject</i> Message was successfully sent.

8.3.2.10.2.3

Data Role Swap, Initiated by UFP Operating as Source (Wait)

Figure 8-47 “Data Role Swap with Wait, UFP operating as Source initiates” shows an example sequence between a Port, which is initially a UFP (Device) and a Source (R_p asserted), and a Port which is initially a DFP (Host) and a Sink (R_d asserted). A Data Role Swap is initiated by the UFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and R_p/R_d remain constant) and the exchange of data roles is delayed with a wait.

Figure 8-47 “Data Role Swap with Wait, UFP operating as Source initiates”

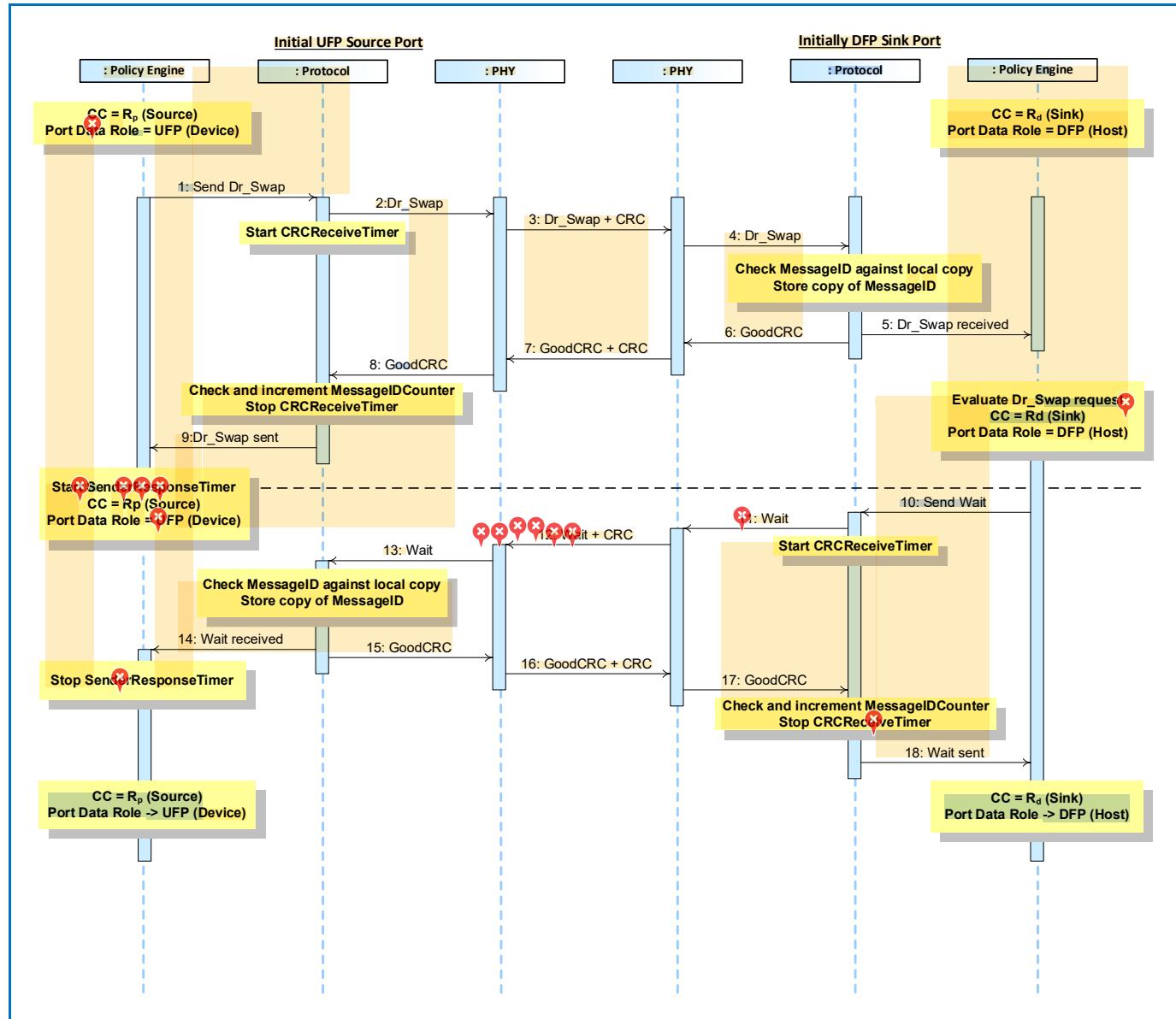


Table 8.75 “Steps for Data Role Swap with Wait, UFP operating as Source initiates” below provides a detailed explanation of what happens at each labeled step in **Figure 8-47 “Data Role Swap with Wait, UFP operating as Source initiates”** above.

Table 8.75 “Steps for Data Role Swap with Wait, UFP operating as Source initiates”

Step	Initial UFP Source Port	Initial DFP Sink Port
1	Port starts as a UFP (Device) operating as Source with R_p asserted and Port Data Role set to UFP. The Policy Engine directs the Protocol Layer to send a DR_Swap Message.	Port starts as a DFP (Host) operating as a Sink with R_d asserted and Port Data Role set to DFP.
2	Protocol Layer creates the DR_Swap Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the DR_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the DR_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the DR_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received DR_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the DR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the DR_Swap Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form a Wait Message.
11		Protocol Layer creates the Wait Message and passes to Physical Layer.
12	Physical Layer receives the Wait Message and checks the CRC to verify the Message.	Physical Layer appends a CRC and sends the Wait Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Wait Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Wait</i> Message was successfully sent.

8.3.2.10.3

Data Role Swap, Initiated by DFP Operating as Source

8.3.2.10.3.1

Data Role Swap, Initiated by DFP Operating as Source (Accept)

Figure 8-48 “Data Role Swap, DFP operating as Source initiates” shows an example sequence between a Port, which is initially a UFP (Device) and a Sink (R_d asserted), and a Port which is initially a DFP and a Source (R_p asserted). A Data Role Swap is initiated by the DFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and R_p/R_d remain constant) but exchange data roles between DFP (Host) and UFP (Device).

Figure 8-48 “Data Role Swap, DFP operating as Source initiates”

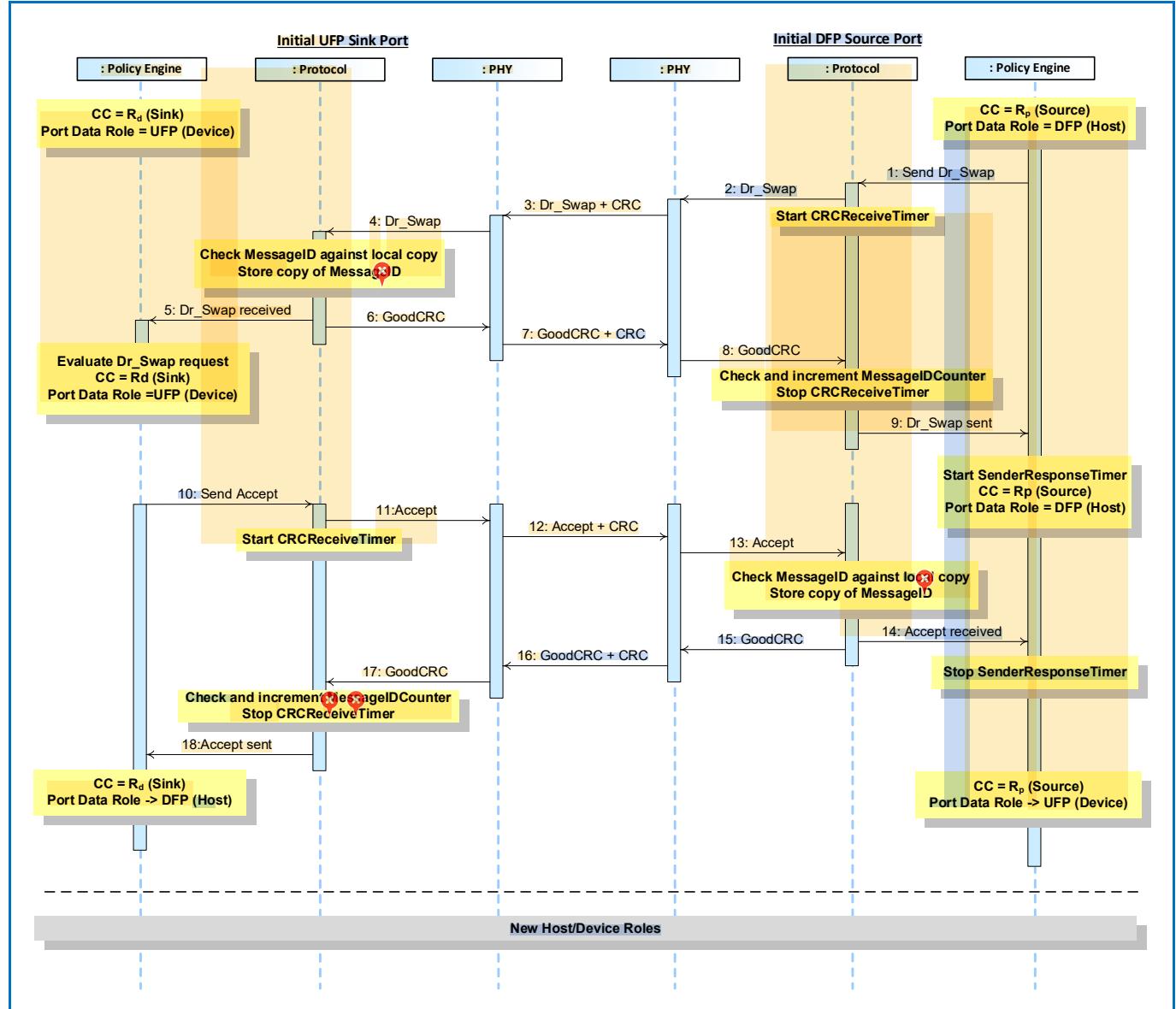


Table 8.76 “Steps for Data Role Swap, DFP operating as Source initiates” below provides a detailed explanation of what happens at each labeled step in **Figure 8-48 “Data Role Swap, DFP operating as Source initiates”** above.

Table 8.76 “Steps for Data Role Swap, DFP operating as Source initiates”

Step	Initial UFP Sink Port	Initial DFP Source Port
1	Port starts as a UFP (Device) operating as a Sink with R_d asserted and Port Data Role set to UFP.	Port starts as a DFP (Host) operating as Source with R_p asserted and Port Data Role set to DFP. The Policy Engine directs the Protocol Layer to send a DR_Swap Message.
2		Protocol Layer creates the DR_Swap Message and passes to Physical Layer.
3	Physical Layer receives the DR_Swap Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the DR_Swap Message. Starts CRCReceiveTimer .
4	Physical Layer removes the CRC and forwards the DR_Swap Message to the Protocol Layer.	
5	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received DR_Swap Message information to the Policy Engine that consumes it.	
6	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
7	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.
8		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
9		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the DR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .
10	Policy Engine evaluates the DR_Swap Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an Accept Message.	
11	Protocol Layer creates the Accept Message and passes to Physical Layer.	
12	Physical Layer appends a CRC and sends the Accept Message. Starts CRCReceiveTimer .	Physical Layer receives the Accept Message and checks the CRC to verify the Message.
13		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it.
14		The Policy Engine stops the SenderResponseTimer .
15		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.

16	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine requests that the Data Role is changed to DFP (Host), with <i>Port Data Role</i> set to DFP, still operating as a Sink (R_d asserted).	The Policy Engine requests that Data Role is changed from DFP (Host) to UFP (Device). The Power Delivery role is now a UFP (Device), with <i>Port Data Role</i> set to UFP and continues supplying power as a Source (R_p asserted).
✖ The Data Role Swap is complete; the data roles have been reversed while maintaining the direction of power flow.		

8.3.2.10.3.2

Data Role Swap, Initiated by DFP Operating as Source (Reject)

Figure 8-49 “Rejected Data Role Swap, DFP operating as Source initiates” shows an example sequence between a Port, which is initially a UFP (Device) and a Sink (R_d asserted), and a Port which is initially a DFP and a Source (R_p asserted). A Data Role Swap is initiated by the DFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and R_p/R_d remain constant) and the exchange of data roles is rejected.

Figure 8-49 “Rejected Data Role Swap, DFP operating as Source initiates”

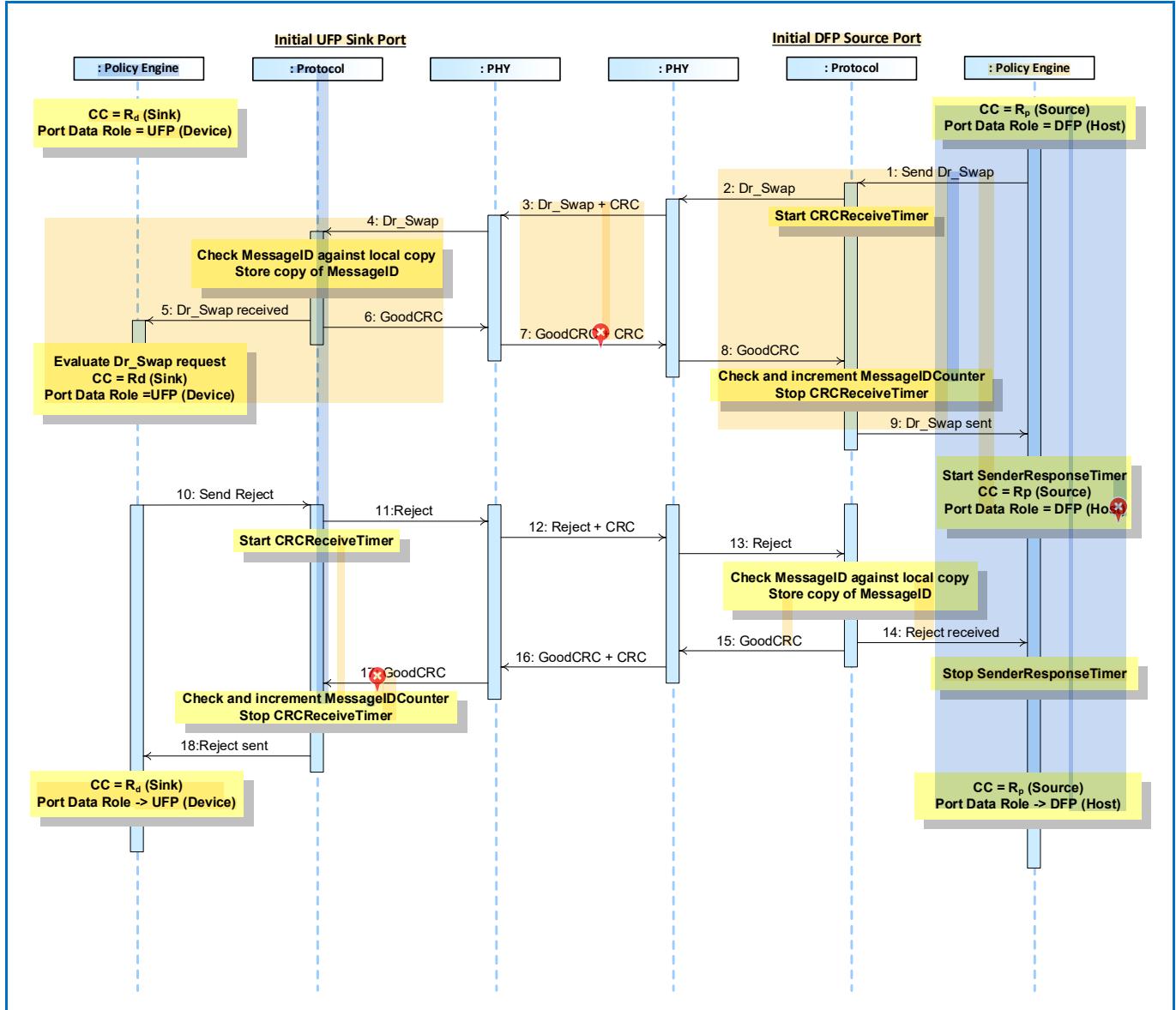


Table 8.77 "Steps for Rejected Data Role Swap, DFP operating as Source initiates" below provides a detailed explanation of what happens at each labeled step in **Figure 8-49 "Rejected Data Role Swap, DFP operating as Source initiates"** above.

 **Table 8.77 "Steps for Rejected Data Role Swap, DFP operating as Source initiates"**

Step	Initial UFP Sink Port	Initial DFP Source Port
1	Port starts as a UFP (Device) operating as a Sink with R_d asserted and Port Data Role set to UFP.	Port starts as a DFP (Host) operating as Source with R_p asserted and Port Data Role set to DFP. The Policy Engine directs the Protocol Layer to send a DR_Swap Message.
2		Protocol Layer creates the DR_Swap Message and passes to Physical Layer.
3	Physical Layer receives the DR_Swap Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the DR_Swap Message. Starts CRCReceiveTimer .
4	Physical Layer removes the CRC and forwards the DR_Swap Message to the Protocol Layer.	
5	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received DR_Swap Message information to the Policy Engine that consumes it.	
6	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
7	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.
8		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
9		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the DR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .
10	Policy Engine evaluates the DR_Swap Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an Reject Message.	
11	Protocol Layer creates the Reject Message and passes to Physical Layer.	
12	Physical Layer appends a CRC and sends the Reject Message. Starts CRCReceiveTimer .	Physical Layer receives the Reject Message and checks the CRC to verify the Message.
13		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Reject Message information to the Policy Engine that consumes it.
14		The Policy Engine stops the SenderResponseTimer .

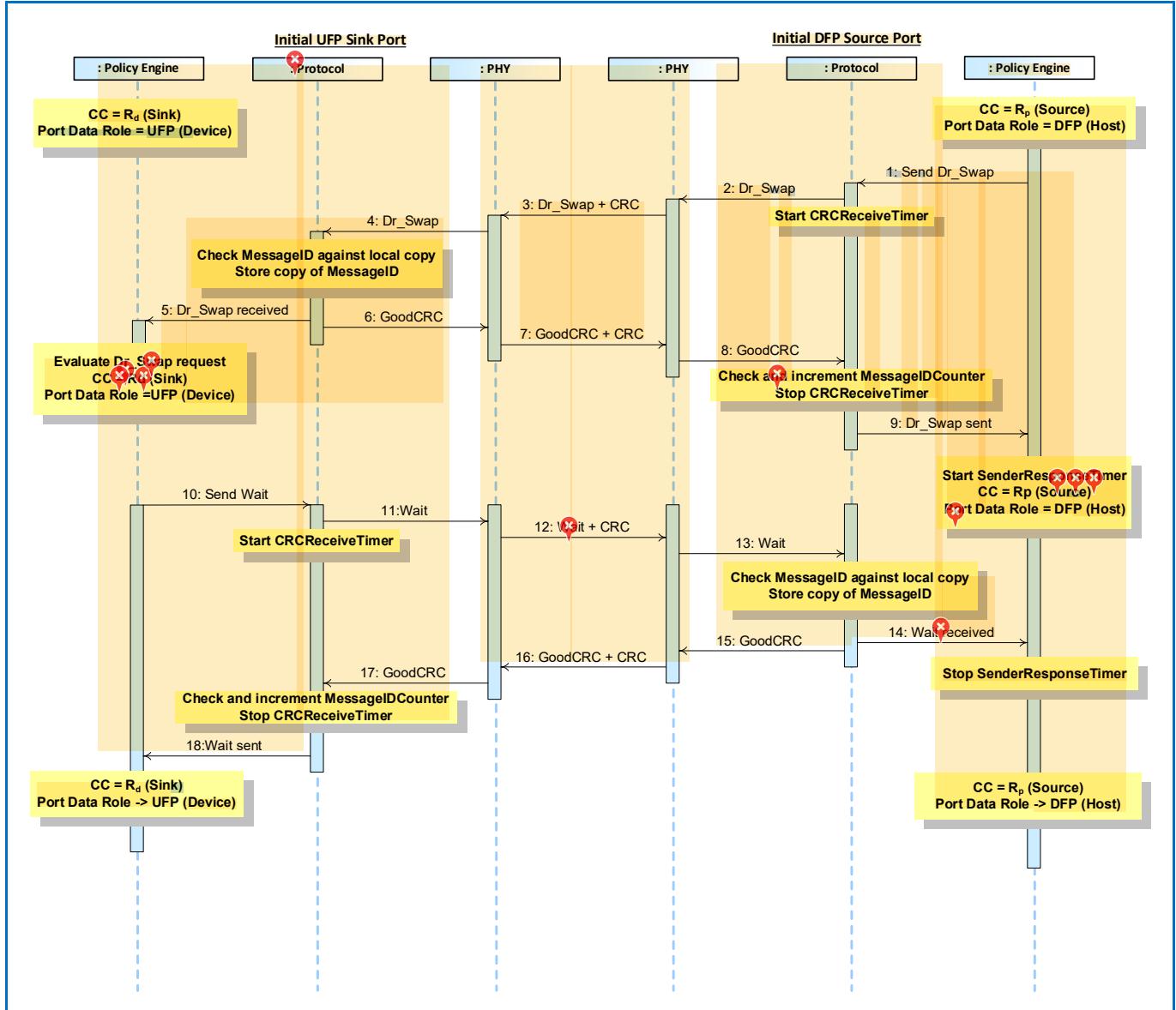
15		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
16	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Reject</i> Message was successfully sent.	

8.3.2.10.3.3

Data Role Swap, Initiated by DFP Operating as Source (Wait)

Figure 8-50 “Data Role Swap with Wait, DFP operating as Source initiates” shows an example sequence between a Port, which is initially a UFP (Device) and a Sink (R_d asserted), and a Port which is initially a DFP and a Source (R_p asserted). A Data Role Swap is initiated by the DFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and R_p/R_d remain constant) and the exchange of data roles is delayed by wait.

Figure 8-50 “Data Role Swap with Wait, DFP operating as Source initiates”



 **Table 8.78 "Steps for Data Role Swap with Wait, DFP operating as Source initiates"** below provides a detailed explanation of what happens at each labeled step in **Figure 8-50 "Data Role Swap with Wait, DFP operating as Source initiates"** above.

 **Table 8.78 "Steps for Data Role Swap with Wait, DFP operating as Source initiates"**

Step	Initial UFP Sink Port	Initial DFP Source Port
1	Port starts as a UFP (Device) operating as a Sink with R_d asserted and Port Data Role set to UFP.	Port starts as a DFP (Host) operating as Source with R_p asserted and Port Data Role set to DFP. The Policy Engine directs the Protocol Layer to send a DR_Swap Message.
2		Protocol Layer creates the DR_Swap Message and passes to Physical Layer.
3	Physical Layer receives the DR_Swap Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the DR_Swap Message. Starts CRCReceiveTimer .
4	Physical Layer removes the CRC and forwards the DR_Swap Message to the Protocol Layer.	
5	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received DR_Swap Message information to the Policy Engine that consumes it.	
6	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
7	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.
8		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
9		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the DR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .
10	Policy Engine evaluates the DR_Swap Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an Wait Message.	
11	Protocol Layer creates the Wait Message and passes to Physical Layer.	
12	Physical Layer appends a CRC and sends the Wait Message. Starts CRCReceiveTimer .	Physical Layer receives the Wait Message and checks the CRC to verify the Message.
13		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Wait Message information to the Policy Engine that consumes it.
14		The Policy Engine stops the SenderResponseTimer .

15		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
16	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Wait</i> Message was successfully sent.	

8.3.2.10.4

Data Role Swap, Initiated by DFP Operating as Sink

8.3.2.10.4.1

Data Role Swap, Initiated by DFP Operating as Sink (Accept)

Figure 8-51 “Data Role Swap, DFP operating as Sink initiates” shows an example sequence between a Port, which is initially a UFP (Device) and a Source (R_p asserted), and a Port which is initially a DFP (Host) and a Sink (R_d asserted). A Data Role Swap is initiated by the DFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and R_p/R_d remain constant) but exchange data roles between DFP (Host) and UFP (Device).

Figure 8-51 “Data Role Swap, DFP operating as Sink initiates”

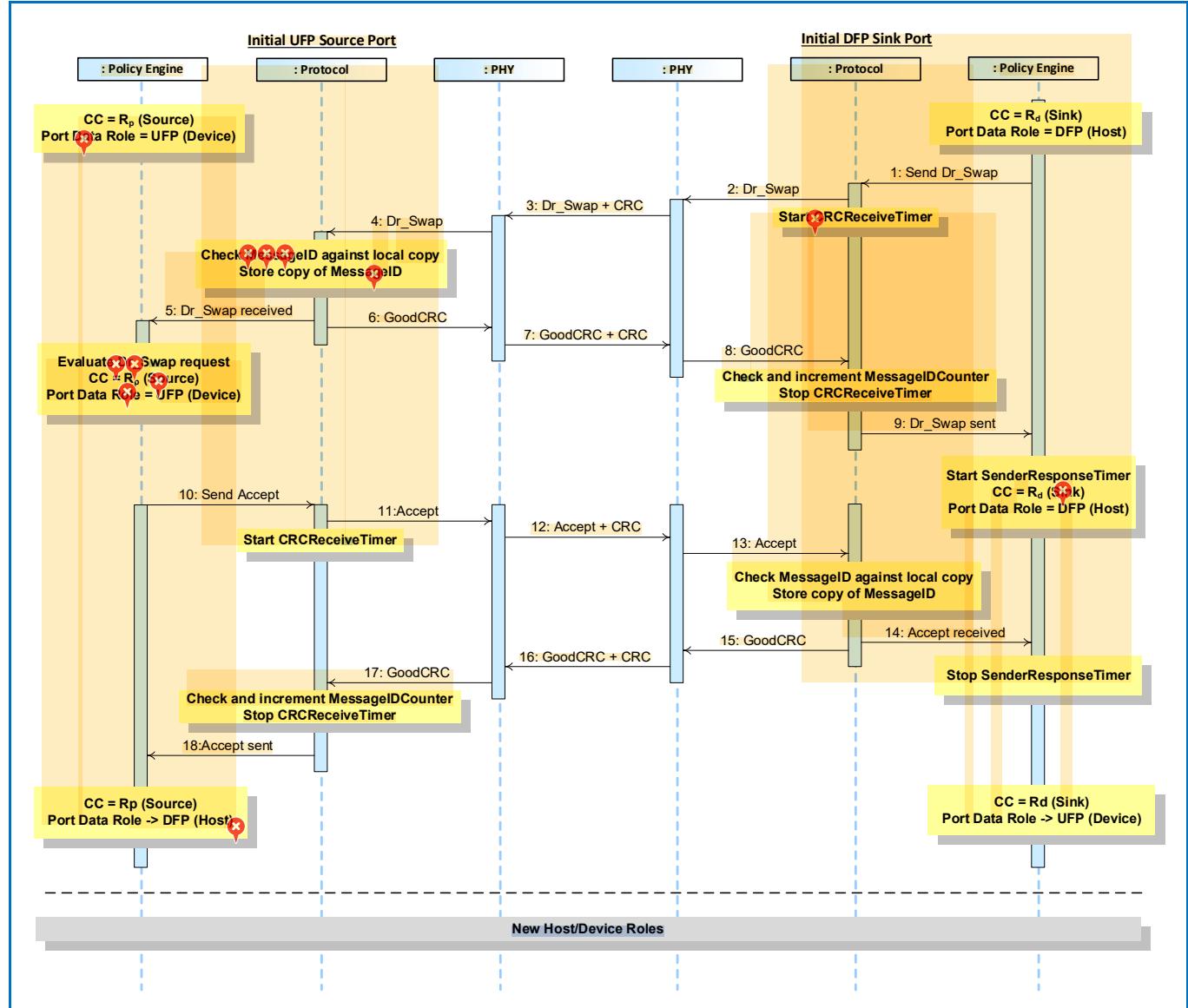


Table 8.79 "Steps for Data Role Swap, DFP operating as Sink initiates" below provides a detailed explanation of what happens at each labeled step in **Figure 8-51 "Data Role Swap, DFP operating as Sink initiates"** above.

Table 8.79 "Steps for Data Role Swap, DFP operating as Sink initiates"

Step	Initial UFP Source Port	Initial DFP Sink Port
1	Port starts as a UFP (Device) operating as Source with R_p asserted and Port Data Role set to UFP.	Port starts as a DFP (Host) operating as a Sink with R_d asserted and Port Data Role set to DFP. The Policy Engine directs the Protocol Layer to send a DR_Swap Message.
2		Protocol Layer creates the DR_Swap Message and passes to Physical Layer.
3	Physical Layer receives the DR_Swap Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the DR_Swap Message. Starts CRCReceiveTimer .
4	Physical Layer removes the CRC and forwards the DR_Swap Message to the Protocol Layer.	
5	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received DR_Swap Message information to the Policy Engine that consumes it.	
6	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
7	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.
8		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
9		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the DR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .
10	Policy Engine evaluates the DR_Swap Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an Accept Message.	
11	Protocol Layer creates the Accept Message and passes to Physical Layer.	
12	Physical Layer appends a CRC and sends the Accept Message. Starts CRCReceiveTimer .	Physical Layer receives the Accept Message and checks the CRC to verify the Message.
13		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it.
14		The Policy Engine stops the SenderResponseTimer .
15		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.

16	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine requests that the Data Role is changed to DFP (Host), with <i>Port Data Role</i> set to UFP and continues supplying power as a Source (R_p asserted).	The Policy Engine requests that Data Role is changed from DFP (Host) to UFP (Device). The Power Delivery role is now a UFP (Device), with <i>Port Data Role</i> set to UFP, still operating as a Sink (R_d asserted).

The Data Role Swap is complete; the data roles have been reversed while maintaining the direction of power flow.

8.3.2.10.4.2

Data Role Swap, Initiated by DFP Operating as Sink (Reject)

Figure 8-52 “Rejected Data Role Swap, DFP operating as Sink initiates” shows an example sequence between a Port, which is initially a UFP (Device) and a Source (R_p asserted), and a Port which is initially a DFP (Host) and a Sink (R_d asserted). A Data Role Swap is initiated by the DFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and R_p/R_d remain constant) and the exchange of data roles is rejected.

Figure 8-52 “Rejected Data Role Swap, DFP operating as Sink initiates”

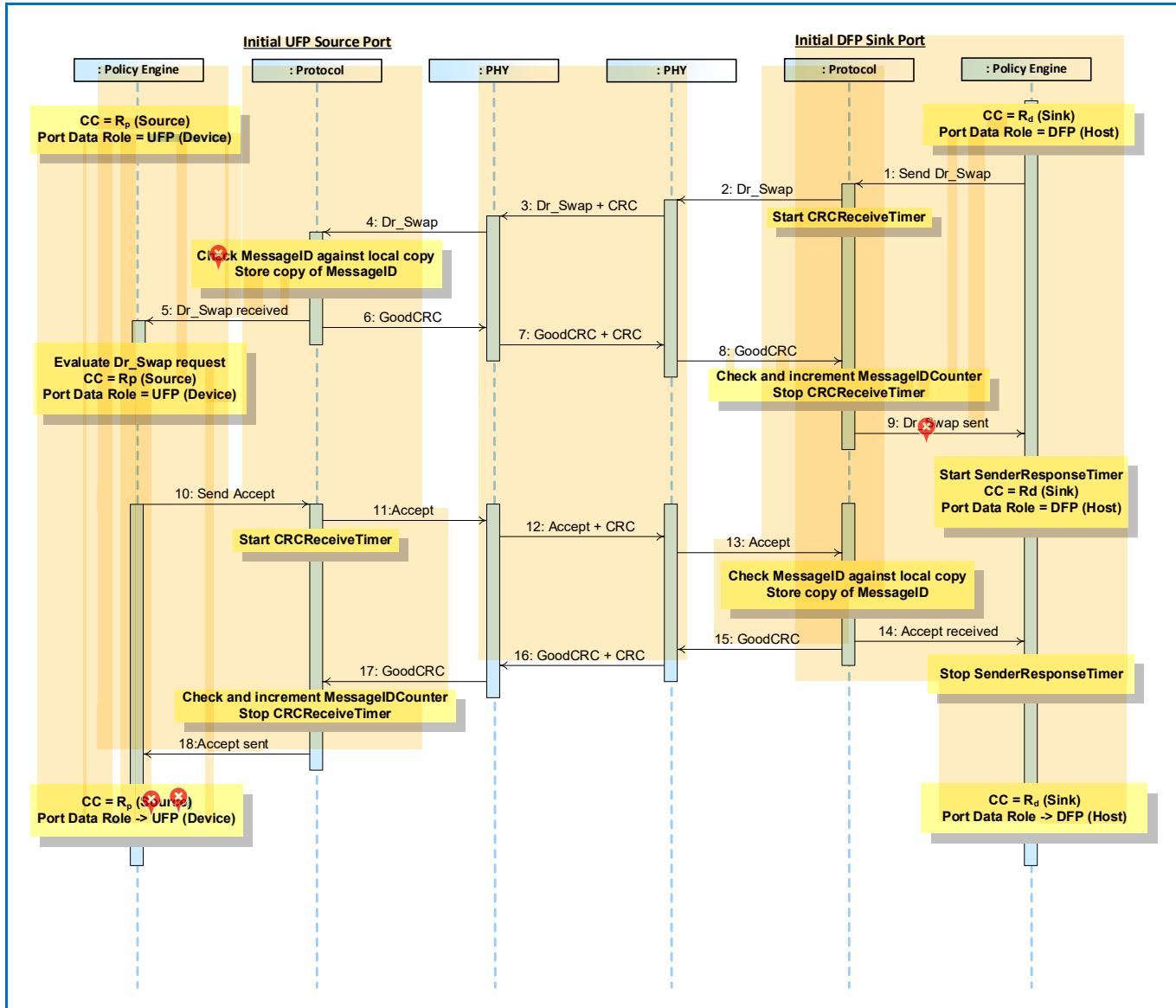


Table 8.80 “Steps for Rejected Data Role Swap, DFP operating as Sink initiates” below provides a detailed explanation of what happens at each labeled step in **Figure 8-52 “Rejected Data Role Swap, DFP operating as Sink initiates”** above.

Table 8.80 “Steps for Rejected Data Role Swap, DFP operating as Sink initiates”

Step	Initial UFP Source Port	Initial DFP Sink Port
1	Port starts as a UFP (Device) operating as Source with R_p asserted and Port Data Role set to UFP.	Port starts as a DFP (Host) operating as a Sink with R_d asserted and Port Data Role set to DFP. The Policy Engine directs the Protocol Layer to send a DR_Swap Message.
2		Protocol Layer creates the DR_Swap Message and passes to Physical Layer.
3	Physical Layer receives the DR_Swap Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the DR_Swap Message. Starts CRCReceiveTimer .
4	Physical Layer removes the CRC and forwards the DR_Swap Message to the Protocol Layer.	
5	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received DR_Swap Message information to the Policy Engine that consumes it.	
6	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
7	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.
8		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
9		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the DR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .
10	Policy Engine evaluates the DR_Swap Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an Reject Message.	
11	Protocol Layer creates the Reject Message and passes to Physical Layer.	
12	Physical Layer appends a CRC and sends the Reject Message. Starts CRCReceiveTimer .	Physical Layer receives the Reject Message and checks the CRC to verify the Message.
13		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Reject Message information to the Policy Engine that consumes it.
14		The Policy Engine stops the SenderResponseTimer .

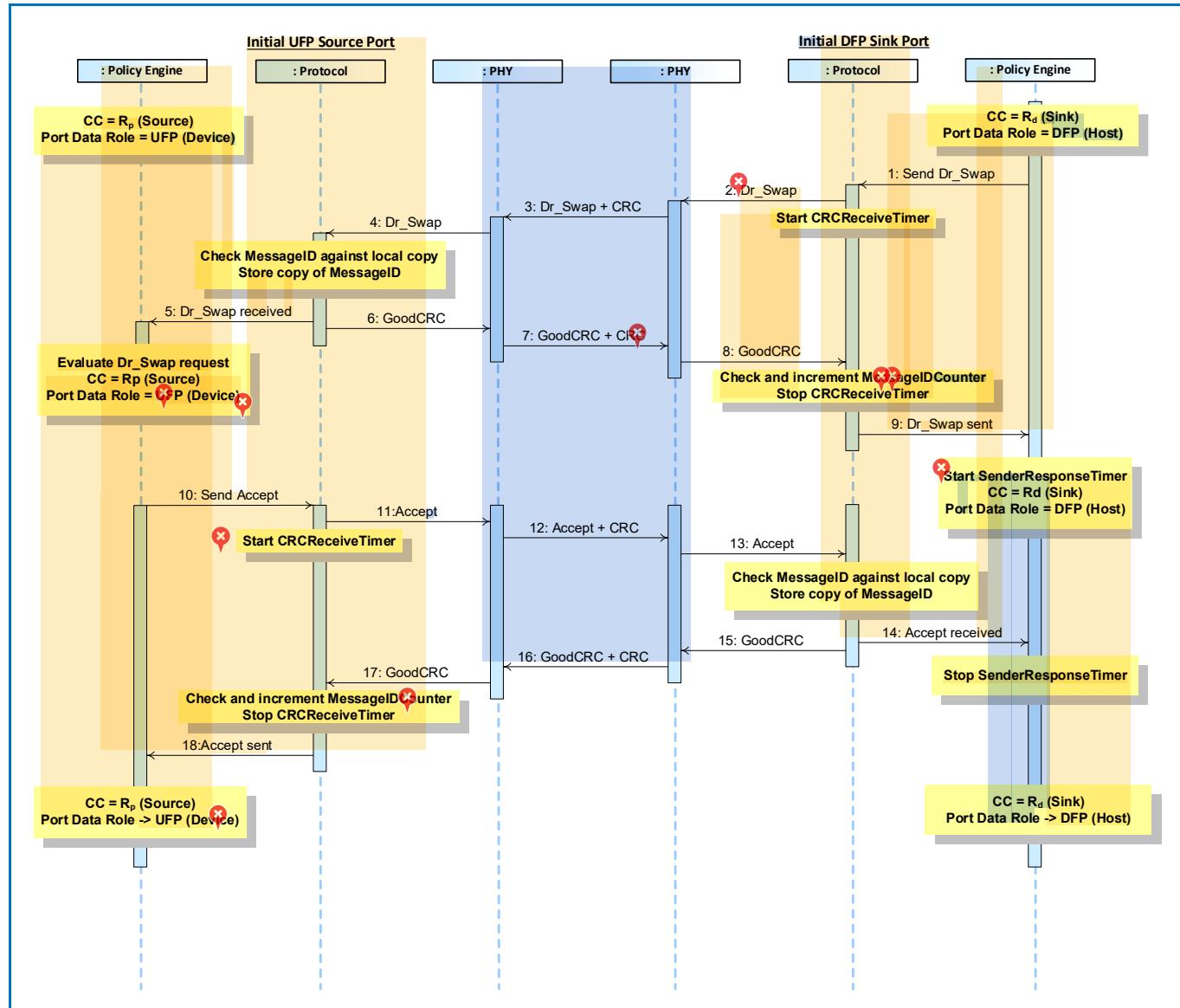
15		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
16	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Reject</i> Message was successfully sent.	

8.3.2.10.4.3

Data Role Swap, Initiated by DFP Operating as Sink (Wait)

Figure 8-53 “Data Role Swap with Wait, DFP operating as Sink initiates” shows an example sequence between a Port, which is initially a UFP (Device) and a Source (R_p asserted), and a Port which is initially a DFP (Host) and a Sink (R_d asserted). A Data Role Swap is initiated by the DFP. During the process the Port Partners maintain their operation as either a Source or a Sink (power and R_p/R_d remain constant) and the exchange of data roles is delayed with a wait.

Figure 8-53 “Data Role Swap with Wait, DFP operating as Sink initiates”



 **Table 8.81 "Steps for Data Role Swap with Wait, DFP operating as Sink initiates"** below provides a detailed explanation of what happens at each labeled step in **Figure 8-53 "Data Role Swap with Wait, DFP operating as Sink initiates"** above.

Table 8.81 "Steps for Data Role Swap with Wait, DFP operating as Sink initiates"

Step	Initial UFP Source Port	Initial DFP Sink Port
1	Port starts as a UFP (Device) operating as Source with R_p asserted and Port Data Role set to UFP.	Port starts as a DFP (Host) operating as a Sink with R_d asserted and Port Data Role set to DFP. The Policy Engine directs the Protocol Layer to send a DR_Swap Message.
2		Protocol Layer creates the DR_Swap Message and passes to Physical Layer.
3	Physical Layer receives the DR_Swap Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the DR_Swap Message. Starts CRCReceiveTimer .
4	Physical Layer removes the CRC and forwards the DR_Swap Message to the Protocol Layer.	
5	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received DR_Swap Message information to the Policy Engine that consumes it.	
6	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
7	Physical Layer appends CRC and sends the GoodCRC Message.	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.
8		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
9		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the DR_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .
10	Policy Engine evaluates the DR_Swap Message and decides that it is able and willing to do the Data Role Swap. It tells the Protocol Layer to form an Reject Message.	
11	Protocol Layer creates the Reject Message and passes to Physical Layer.	
12	Physical Layer appends a CRC and sends the Reject Message. Starts CRCReceiveTimer .	Physical Layer receives the Reject Message and checks the CRC to verify the Message.
13		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Reject Message information to the Policy Engine that consumes it.
14		The Policy Engine stops the SenderResponseTimer .

15		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
16	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Reject</i> Message was successfully sent.	

8.3.2.11 VCONN Swap

8.3.2.11.1 VCONN Source Swap, initiated by VCONN Source

8.3.2.11.1.1 VCONN Source Swap, initiated by VCONN Source (Accept)

Figure 8-54 “Successful Vconn Source Swap, initiated by Vconn Source” shows an example sequence where the VCONN Source and tells its Port Partner to supply VCONN. During the process the Port Partners, keep their role as Source or Sink, maintain their operation as either a Source or a Sink (power remains constant) but exchange the VCONN Source role.

Figure 8-54 “Successful VCONN Source Swap, initiated by VCONN Source”

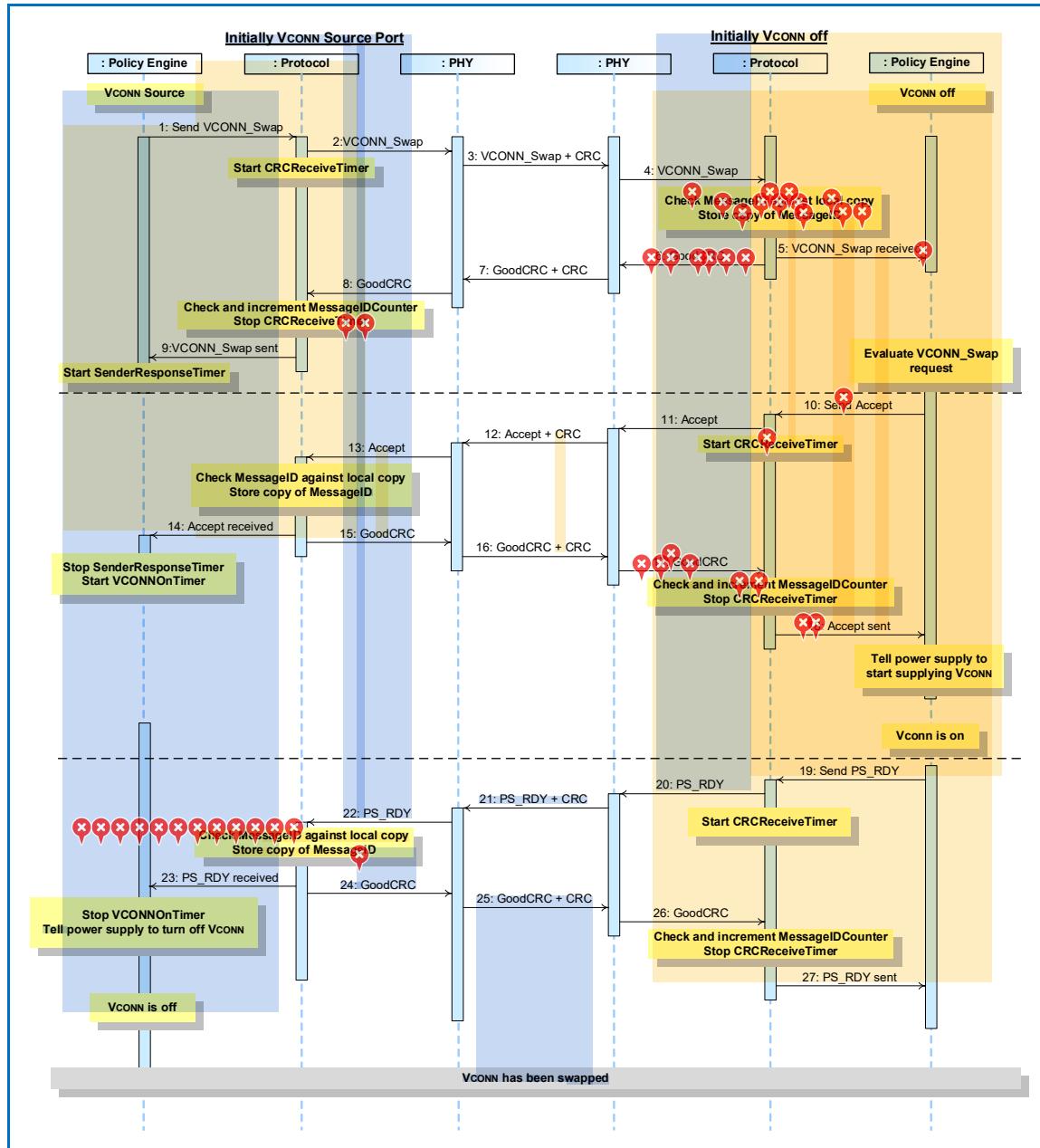


Table 8.82 “Steps for Source to Sink Vconn Source Swap” below provides a detailed explanation of what happens at each labeled step in **Figure 8-54 “Successful Vconn Source Swap, initiated by Vconn Source”** above.

📍 **Table 8.82 “Steps for Source to Sink VCONN Source Swap”**

Step	Initially VCONN Source	Initially VCONN off
1	The VCONN Source’s Policy Engine directs the Protocol Layer to send a VCONN_Swap Message.	VCONN is off.
2	Protocol Layer creates the VCONN_Swap Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the VCONN_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the VCONN_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the VCONN_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received VCONN_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the VCONN_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the VCONN_Swap Message sent by the Source and decides that it is able and willing to do the VCONN Swap. It tells the Protocol Layer to form an Accept Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Accept Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Accept Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer and starts the VCONNOOnTimer .	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	

16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent. The Policy Engine asks the Device Policy Manager to turn on VCONN.
19		The Device Policy Manager informs the Policy Engine that its power supply is supplying VCONN. The Policy Engine directs the Protocol Layer to generate a <i>PS_RDY</i> Message to tell the Source it can turn off VCONN.
20		Protocol Layer creates the <i>PS_RDY</i> Message and passes to Physical Layer.
21	Physical Layer receives the <i>PS_RDY</i> Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>PS_RDY</i> Message. Starts <i>CRCReceiveTimer</i> .
22	Physical Layer removes the CRC and forwards the <i>PS_RDY</i> Message to the Protocol Layer.	
23	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>PS_RDY</i> Message information to the Policy Engine that consumes it.	
24	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
25	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message. The Policy Engine stops the <i>VCONNOnTimer</i> , and tells the power supply to stop sourcing VCONN.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
26		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
27	VCONN is off.	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PS_RDY</i> Message was successfully sent.

The Ports have swapped VCONN Source role.

8.3.2.11.1.2

VCONN Source Swap, initiated by VCONN Source (Reject)

Figure 8-55 “Rejected Vconn Source Swap, initiated by Vconn Source” shows an example sequence where the VCONN Source and tells its Port Partner to supply VCONN and is rejected. During the process the Port Partners, keep their role as Source or Sink, maintain their operation as either a Source or a Sink (power remains constant) and don't exchange the VCONN Source role.

Figure 8-55 “Rejected VCONN Source Swap, initiated by VCONN Source”

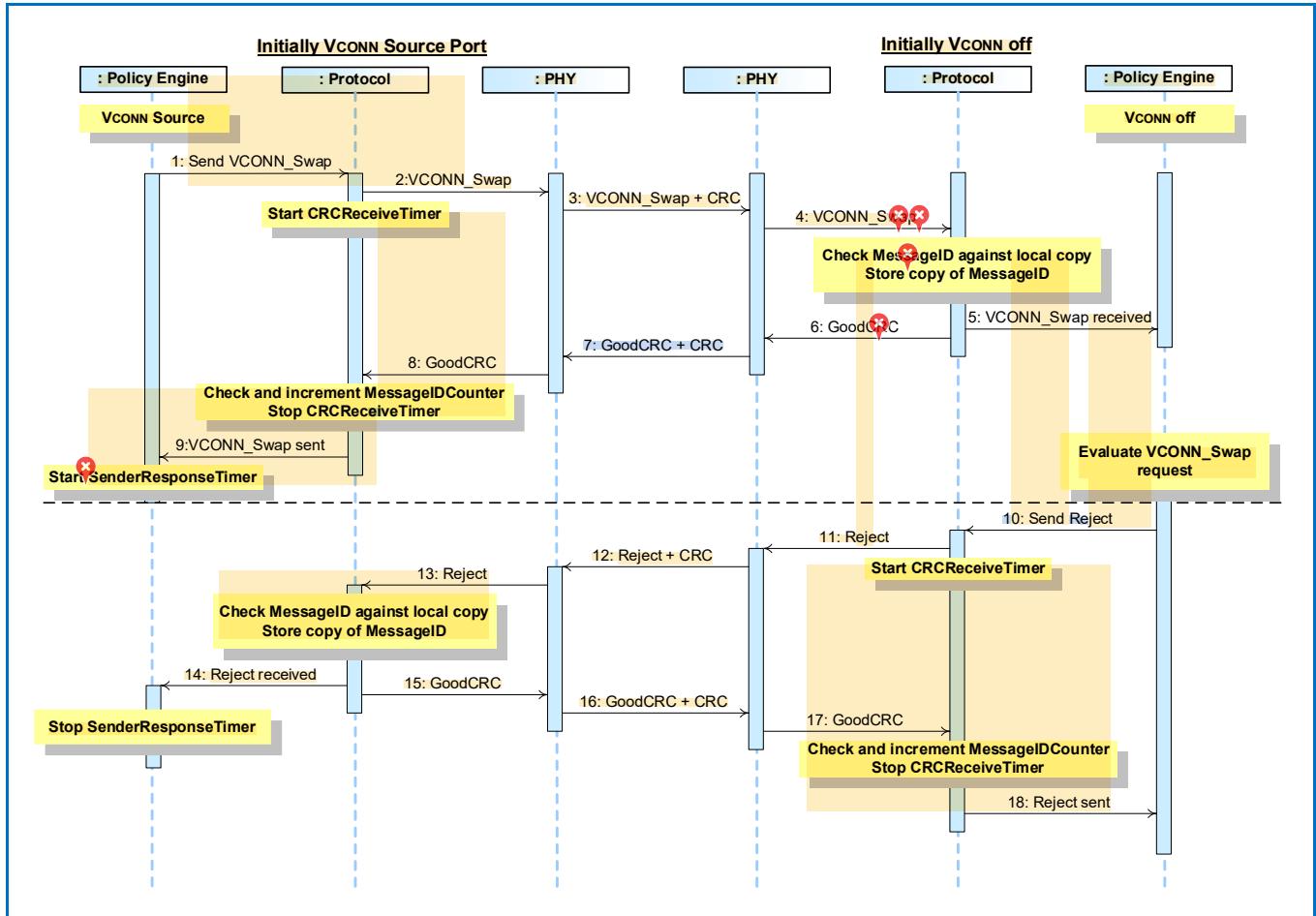


Table 8.83 "Steps for Rejected Vconn Source Swap" below provides a detailed explanation of what happens at each labeled step in **Figure 8-55 "Rejected Vconn Source Swap, initiated by Vconn Source"** above.

Table 8.83 "Steps for Rejected VCONN Source Swap"

Step	Initially VCONN Source	Initially VCONN off
1	The VCONN Source's Policy Engine directs the Protocol Layer to send a VCONN_Swap Message.	VCONN is off.
2	Protocol Layer creates the VCONN_Swap Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the VCONN_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the VCONN_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the VCONN_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received VCONN_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the VCONN_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the VCONN_Swap Message sent by the Source and decides that it is able and willing to do the VCONN Swap. It tells the Protocol Layer to form an Reject Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Reject Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Reject Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Reject Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer and starts the VCONNOOnTimer .	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	

16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Reject</i> Message was successfully sent

8.3.2.11.1.3

VCONN Source Swap, initiated by VCONN Source (Wait)

Figure 8-56 “Vconn Source Swap with Wait, initiated by Vconn Source” shows an example sequence where the VCONN Source and tells its Port Partner to supply VCONN and is told to wait. During the process the Port Partners, keep their role as Source or Sink, maintain their operation as either a Source or a Sink (power remains constant) and don’t exchange the VCONN Source role.

Figure 8-56 “VCONN Source Swap with Wait, initiated by VCONN Source”

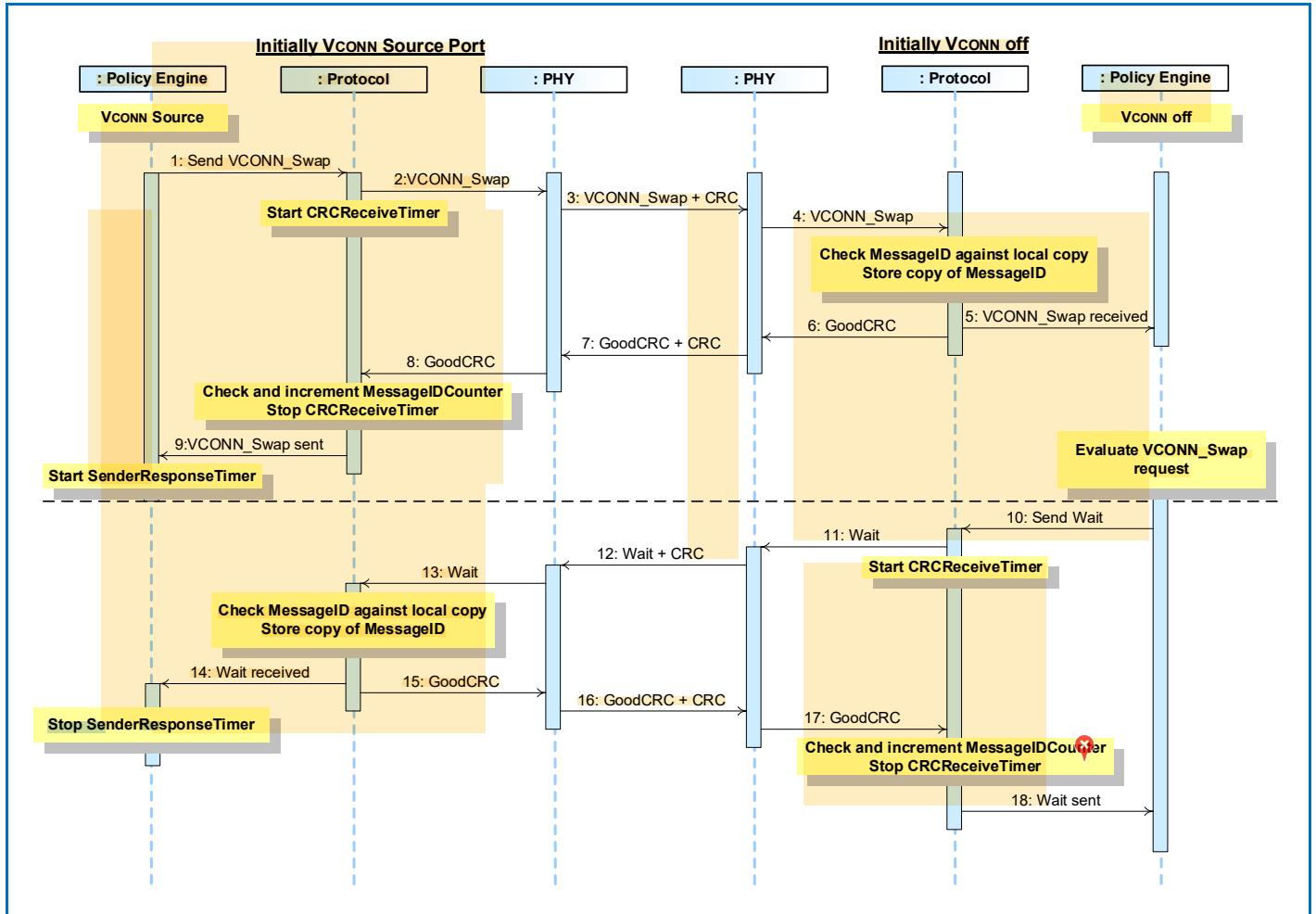


Table 8.84 "Steps for Vconn Source Swap with Wait" below provides a detailed explanation of what happens at each labeled step in **Figure 8-56 "Vconn Source Swap with Wait, initiated by Vconn Source"** above.

Table 8.84 "Steps for VCONN Source Swap with Wait"

Step	Initially VCONN Source	Initially VCONN off
1	The VCONN Source's Policy Engine directs the Protocol Layer to send a VCONN_Swap Message.	VCONN is off.
2	Protocol Layer creates the VCONN_Swap Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the VCONN_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the VCONN_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the VCONN_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received VCONN_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the VCONN_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the VCONN_Swap Message sent by the Source and decides that it is able and willing to do the VCONN Swap. It tells the Protocol Layer to form an Wait Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Wait Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Wait Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Wait Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer and starts the VCONNOOnTimer .	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	

16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Wait</i> Message was successfully sent

8.3.2.11.2 VCONN Source Swap, initiated by non- VCONN Source

8.3.2.11.2.1 VCONN Source Swap, initiated by non- VCONN Source (Accept)

Figure 8-57 “Vconn Source Swap, initiated by non- Vconn Source” shows an example where the Port which is not initially supplying VCONN and requests a VCONN Swap. During the process the Port Partners, keep their role as Source or Sink, maintain their operation as either a Source or a Sink (power remains constant) but exchange the VCONN Source.

Figure 8-57 “VCONN Source Swap, initiated by non- VCONN Source”

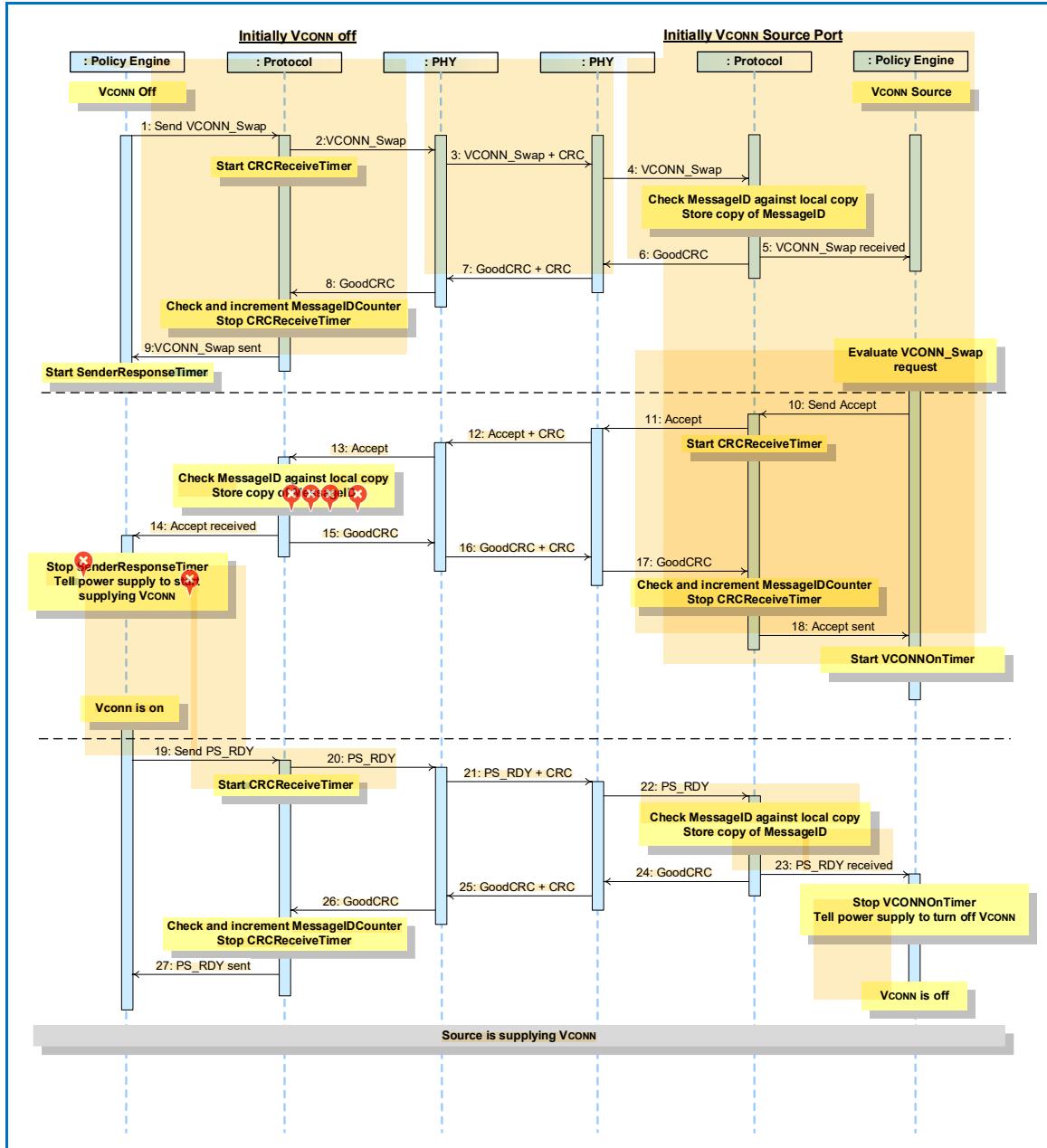


Table 8.85 “Steps for Vconn Source Swap, Initiated by non- Vconn Source” below provides a detailed explanation of what happens at each labeled step in **Figure 8-57 “Vconn Source Swap, initiated by non- Vconn Source”** above.

Table 8.85 “Steps for VCONN Source Swap, Initiated by non- VCONN Source”

Step	Initially VCONN off	Initially VCONN Source
1	The Source starts with VCONN off. The Policy Engine directs the Protocol Layer to send a VCONN_Swap Message.	The Sink starts as the VCONN Source.
2	Protocol Layer creates the VCONN_Swap Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the VCONN_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the VCONN_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the VCONN_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received VCONN_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the VCONN_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the VCONN_Swap Message sent by the Source and decides that it is able and willing to do the VCONN Swap. It tells the Protocol Layer to form an Accept Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Accept Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Accept Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer . The Policy Engine tells the Device Policy Manager to turn on VCONN.	

15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
18		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Accept Message was successfully sent. The Policy Engine starts the VCONNOnTimer .
19	The Device Policy Manager tells the Policy Engine that its power supply is supplying VCONN. The Policy Engine directs the Protocol Layer to generate a PS_RDY Message to tell the Sink it can turn off VCONN.	
20	Protocol Layer creates the PS_RDY Message and passes to Physical Layer.	
21	Physical Layer appends a CRC and sends the PS_RDY Message. Starts CRCReceiveTimer .	Physical Layer receives the PS_RDY Message and compares the CRC it calculated with the one sent to verify the Message.
22		Physical Layer removes the CRC and forwards the PS_RDY Message to the Protocol Layer.
23		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received PS_RDY Message information to the Policy Engine that consumes it.
24		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
25	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the GoodCRC Message. The Policy Engine stops the VCONNOnTimer , and tells the power supply to stop sourcing VCONN.
26	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
27	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the PS_RDY Message was successfully sent.	VCONN is off.
The Ports have swapped VCONN Source role.		

8.3.2.11.2.2

VCONN Source Swap, initiated by non- VCONN Source (Reject)

Figure 8-58 “Rejected Vconn Source Swap, initiated by non- Vconn Source” shows an example where the Port which is not initially supplying VCONN and requests a VCONN Swap which is rejected. During the process the Port Partners, keep their role as Source or Sink, maintain their operation as either a Source or a Sink (power remains constant) and don't exchange the VCONN Source.

Figure 8-58 “Rejected VCONN Source Swap, initiated by non- VCONN Source”

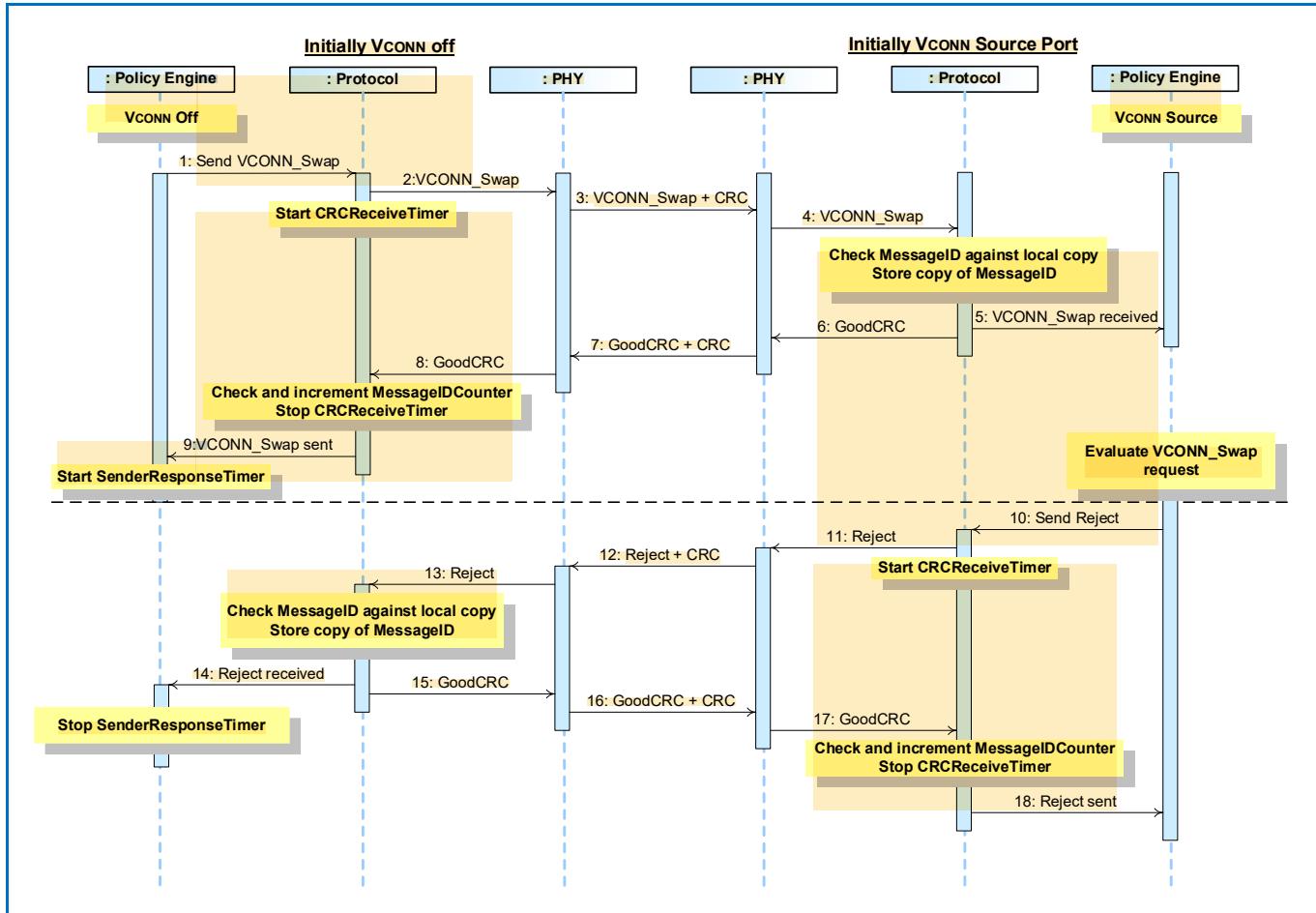


Table 8.86 “Steps for Rejected Vconn Source Swap, Initiated by non- Vconn Source” below provides a detailed explanation of what happens at each labeled step in **Figure 8-58 “Rejected Vconn Source Swap, initiated by non- Vconn Source”** above.

Table 8.86 “Steps for Rejected VCONN Source Swap, Initiated by non- VCONN Source”

Step	Initially VCONN off	Initially VCONN Source
1	The Source starts with VCONN off. The Policy Engine directs the Protocol Layer to send a VCONN_Swap Message.	The Sink starts as the VCONN Source.
2	Protocol Layer creates the VCONN_Swap Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the VCONN_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the VCONN_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the VCONN_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received VCONN_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the VCONN_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the VCONN_Swap Message sent by the Source and decides that it is able and willing to do the VCONN Swap. It tells the Protocol Layer to form a Reject Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Reject Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Reject Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Reject Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer . The Policy Engine tells the Device Policy Manager to turn on VCONN.	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Reject</i> Message was successfully sent.



8.3.2.11.2.3

VCONN Source Swap, initiated by non- VCONN Source (Wait)

Figure 8-59 “Vconn Source Swap with Wait, initiated by non- Vconn Source” shows an example where the Port which is not initially supplying VCONN and requests a VCONN Swap which is delayed with a wait. During the process the Port Partners, keep their role as Source or Sink, maintain their operation as either a Source or a Sink (power remains constant) and don't exchange the VCONN Source.

Figure 8-59 “VCONN Source Swap with Wait, initiated by non- VCONN Source”

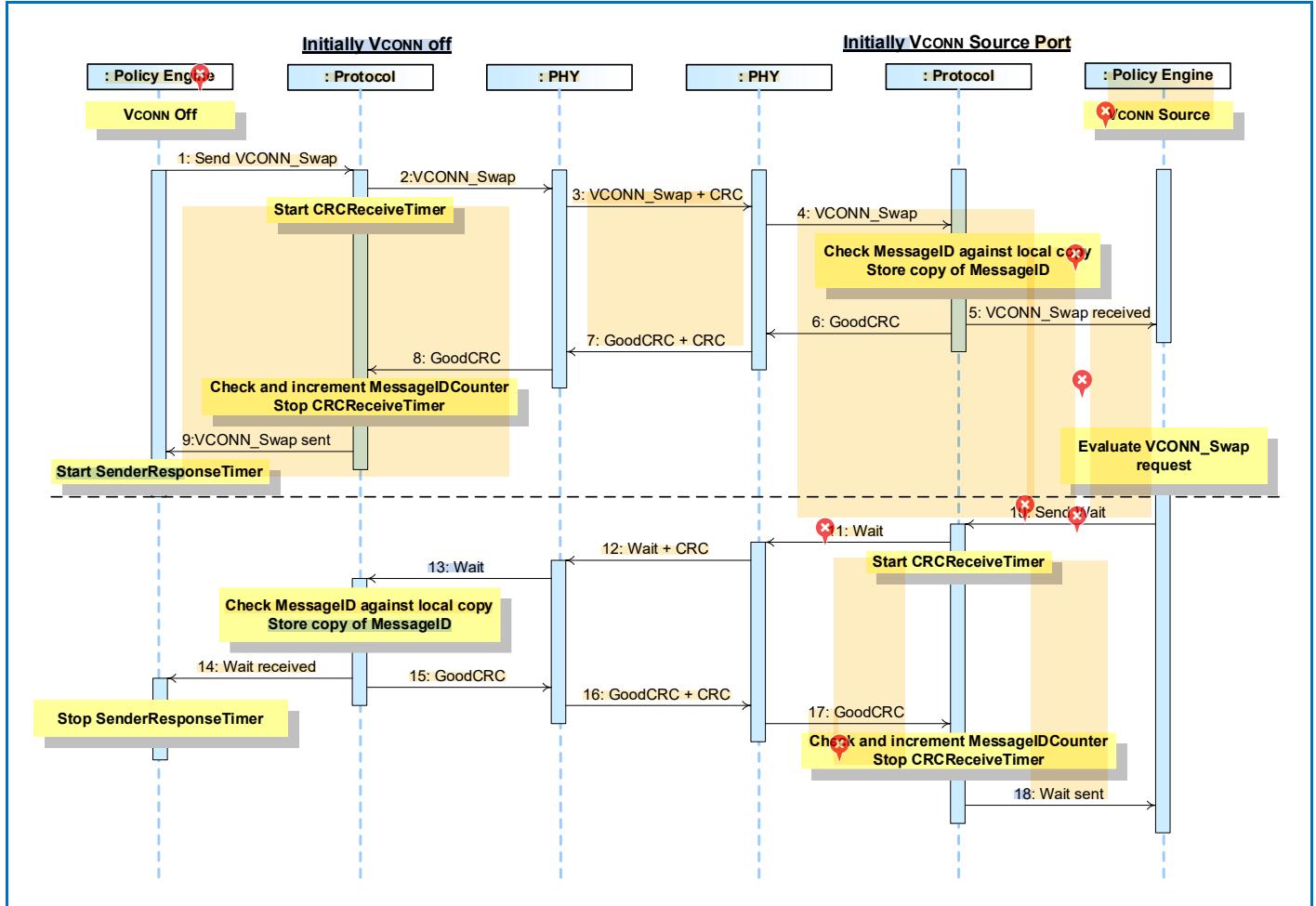


Table 8.87 "Steps for Vconn Source Swap with Wait, Initiated by non- Vconn Source" below provides a detailed explanation of what happens at each labeled step in **Figure 8-59 "Vconn Source Swap with Wait, initiated by non-Vconn Source"** above.

Table 8.87 "Steps for VCONN Source Swap with Wait, Initiated by non- VCONN Source"

Step	Initially VCONN off	Initially VCONN Source
1	The Source starts with VCONN off. The Policy Engine directs the Protocol Layer to send a VCONN_Swap Message.	The Sink starts as the VCONN Source.
2	Protocol Layer creates the VCONN_Swap Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the VCONN_Swap Message. Starts CRCReceiveTimer .	Physical Layer receives the VCONN_Swap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the VCONN_Swap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received VCONN_Swap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the VCONN_Swap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine evaluates the VCONN_Swap Message sent by the Source and decides that it is able and willing to do the VCONN Swap. It tells the Protocol Layer to form a Wait Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Wait Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Wait Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Wait Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer . The Policy Engine tells the Device Policy Manager to turn on VCONN.	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Wait</i> Message was successfully sent.

8.3.2.12 Additional Capabilities, Status and Information

8.3.2.12.1 Alert

8.3.2.12.1.1 Source sends Alert to a Sink

Figure 8-60 “Source Alert to Sink” shows an example sequence between a Source and a Sink where the Source alerts the Sink that there has been a status change. This AMS will be followed by getting the Source status to determine further details of the alert (see [Section 8.3.2.12.2 “Status”](#)).

Figure 8-60 “Source Alert to Sink”

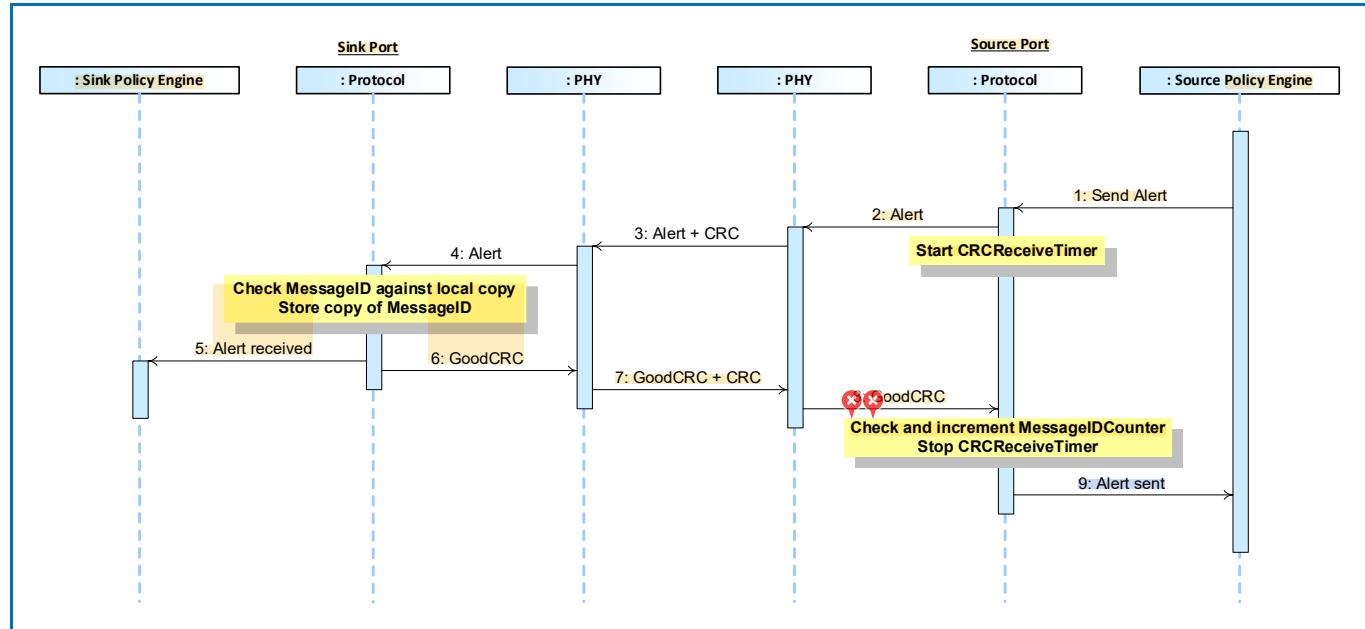


Table 8.88 “Steps for Source Alert to Sink” below provides a detailed explanation of what happens at each labeled step in **Figure 8-60 “Source Alert to Sink”** above.

Table 8.88 “Steps for Source Alert to Sink”

Step	Sink	Source
1		The Device Policy Manager indicates a Source alert condition. The Policy Engine tells the Protocol Layer to form an Alert Message.
2		Protocol Layer creates the Alert Message and passes to Physical Layer.
3	Physical Layer receives the Alert Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Alert Message. Starts CRCReceiveTimer .
4	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Alert Message to the Policy Engine that consumes it.	
5	The Policy Engine informs the Device Policy Manager.	
6	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
7	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
8		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
9		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Alert Message was successfully sent.

8.3.2.12.1.2

Sink sends Alert to a Source

Figure 8-61 “Sink Alert to Source” shows an example sequence between a Source and a Sink where the Sink alerts the Source that there has been a status change. This AMS will be followed by getting the Sink status to determine further details of the alert (see [Section 8.3.2.12.2 “Status”](#)).

Figure 8-61 “Sink Alert to Source”

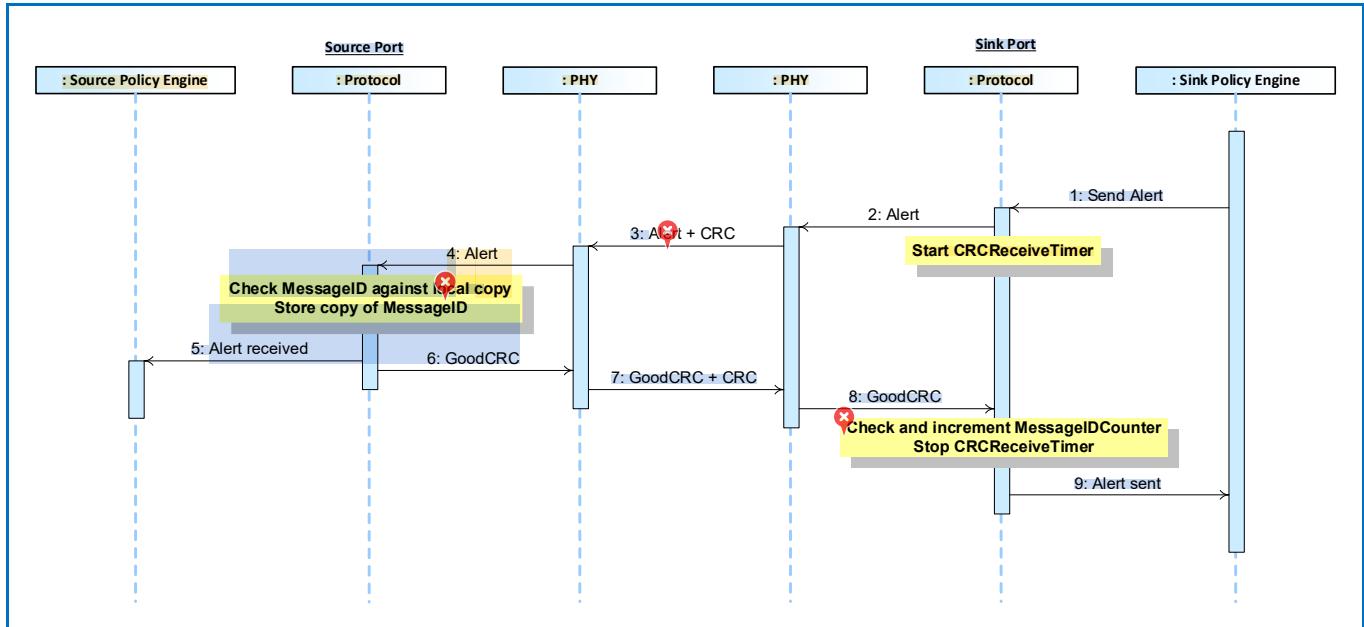


Table 8.89 “Steps for Sink Alert to Source” below provides a detailed explanation of what happens at each labeled step in **Figure 8-61 “Sink Alert to Source”** above.

Table 8.89 “Steps for Sink Alert to Source”

Step	Source	Sink
1		The Device Policy Manager indicates a Sink alert condition. The Policy Engine tells the Protocol Layer to form an Alert Message.
2		Protocol Layer creates the Alert Message and passes to Physical Layer.
3	Physical Layer receives the Alert Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Alert Message. Starts CRCReceiveTimer .
4	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Alert Message to the Policy Engine that consumes it.	
5	The Policy Engine informs the Device Policy Manager.	
6	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
7	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
8		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
9		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Alert Message was successfully sent.

8.3.2.12.2 Status

8.3.2.12.2.1 Sink Gets Source Status

Figure 8-62 “Sink Gets Source Status” shows an example sequence between a Source and a Sink where, after the Sink has received an alert (see [Section 8.3.2.12.2 “Status”](#)) that there has been a status change, the Sink gets more details on the change.

Figure 8-62 “Sink Gets Source Status”

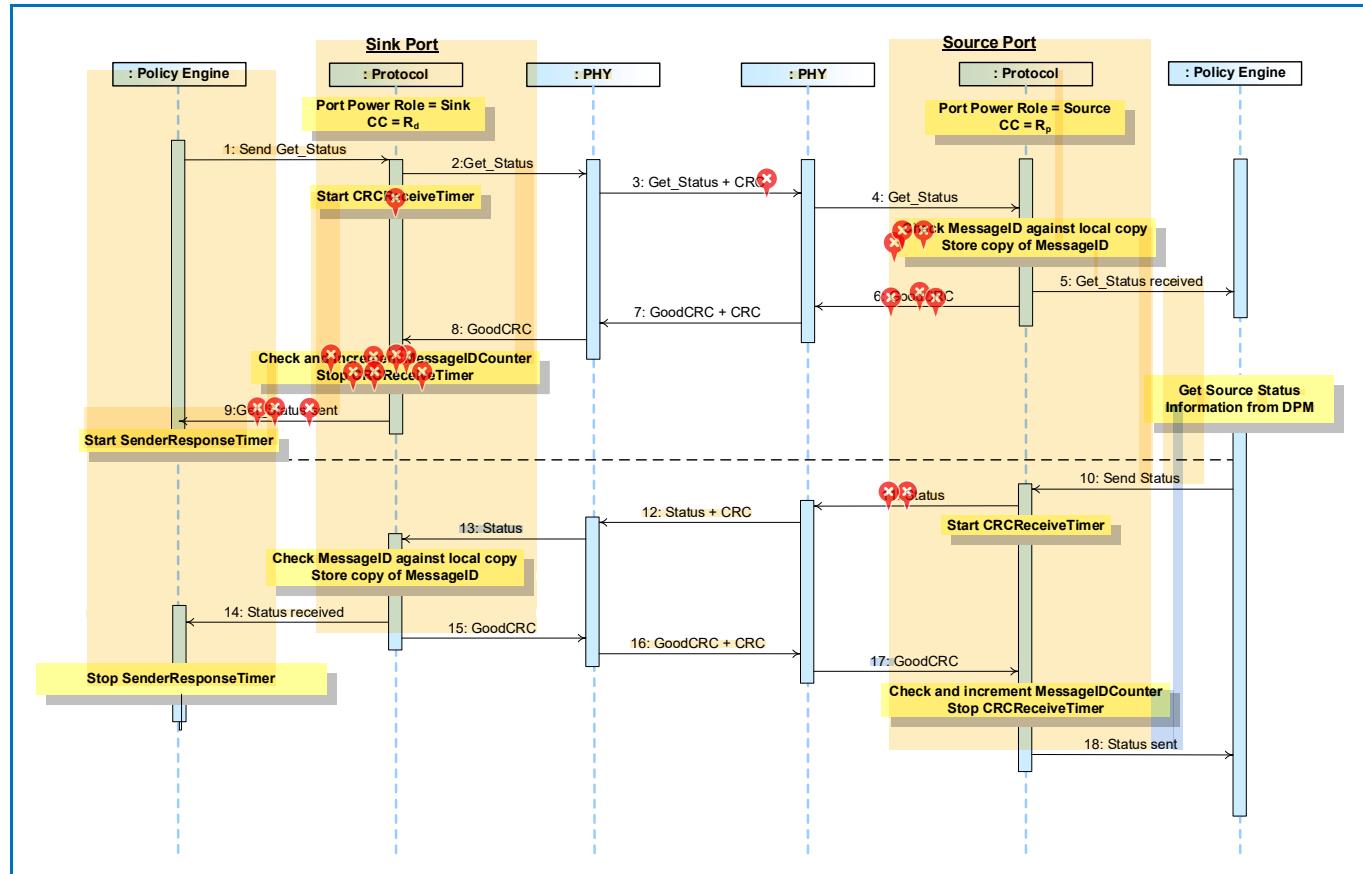


Table 8.90 "Steps for a Sink getting Source Status Sequence" below provides a detailed explanation of what happens at each labeled step in **Figure 8-62 "Sink Gets Source Status"** above.

Table 8.90 "Steps for a Sink getting Source Status Sequence"

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Status Message.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Status Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Status Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Status Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Status Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Status Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Source status which is provided. The Policy Engine tells the Protocol Layer to form a Status Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Status Message.	Physical Layer appends a CRC and sends the Status Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Status Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC?</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Status?</i> Message was successfully sent.
The Source has informed the Sink of its present status.		

8.3.2.12.2.2

Source Gets Sink Status

Figure 8-63 “Source Gets Sink Status” shows an example sequence between a Source and a Sink where, after the Source has received an alert (see [Section 8.3.2.12.2 “Status”](#)) that there has been a status change, the Source gets more details on the change.

Figure 8-63 “Source Gets Sink Status”

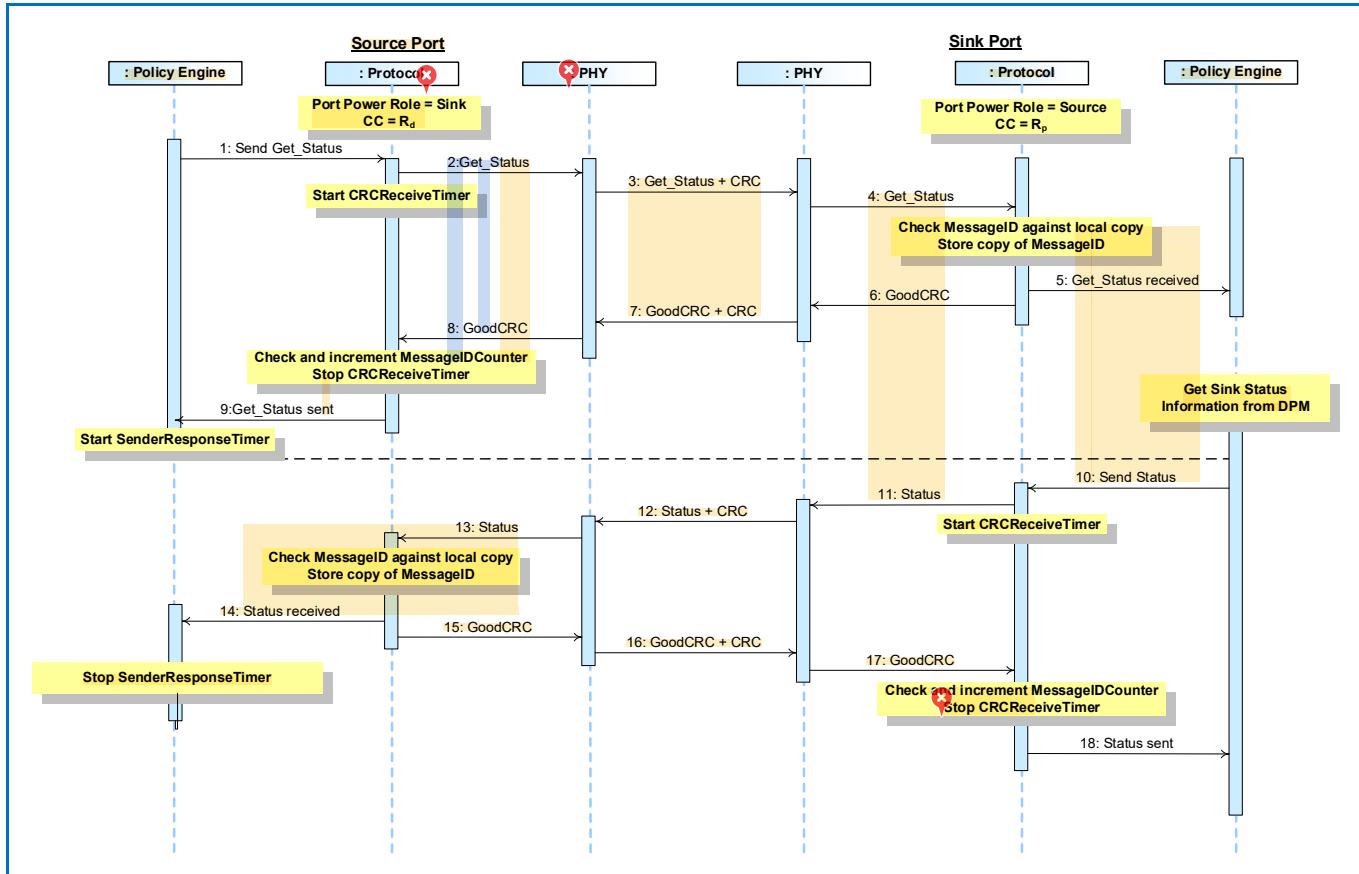


Table 8.91 “Steps for a Source getting Sink Status Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-63 “Source Gets Sink Status”** above.

Table 8.91 “Steps for a Source getting Sink Status Sequence”

Step	Source Port	Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Status Message.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Status Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Status Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Status Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Status Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Status Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Source status which is provided. The Policy Engine tells the Protocol Layer to form a Status Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Status Message.	Physical Layer appends a CRC and sends the Status Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Status Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Status</i> Message was successfully sent.
The Sink has informed the Source of its present status.		

8.3.2.12.2.3

VCONN Source Gets Cable Plug Status

Figure 8-64 “Vconn Source Gets Cable Plug Status” shows an example sequence between a VCONN Source and a Cable Plug where, after the VCONN Source has received an alert (see [Section 8.3.2.12.2 “Status”](#)) that there has been a status change, the VCONN Source gets more details on the change.

Figure 8-64 “VCONN Source Gets Cable Plug Status”

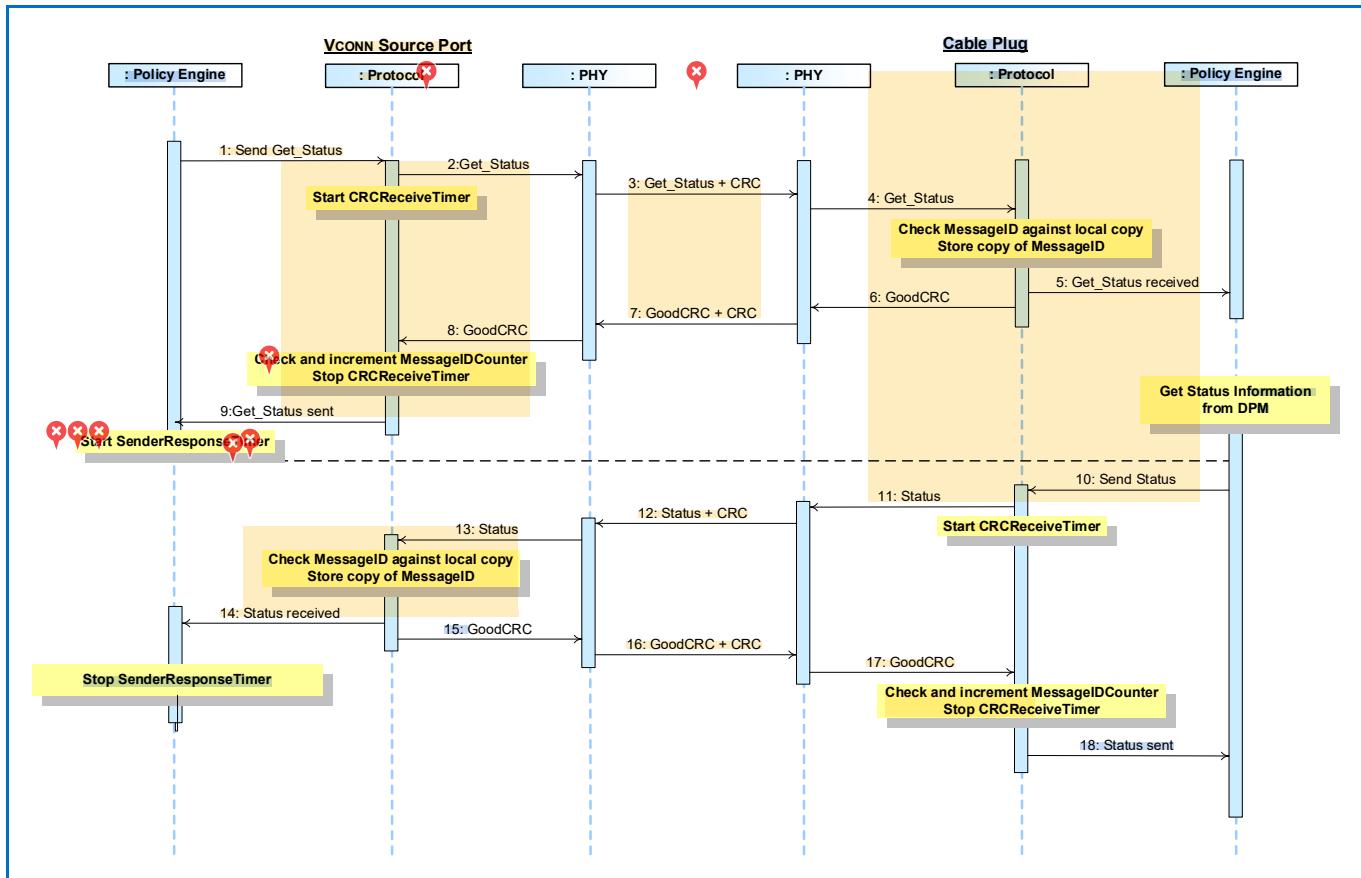


Table 8.92 “Steps for a Vconn Source getting Cable Plug Status Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-64 “Vconn Source Gets Cable Plug Status”** above.

Table 8.92 “Steps for a VCONN Source getting Cable Plug Status Sequence”

Step	VCONN Source Port	Cable Plug
1	Policy Engine directs the Protocol Layer to send a Get_Status Message.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Status Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Status Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Status Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Status Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Status Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Source status which is provided. The Policy Engine tells the Protocol Layer to form a Status Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Status Message.	Physical Layer appends a CRC and sends the Status Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Status Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	

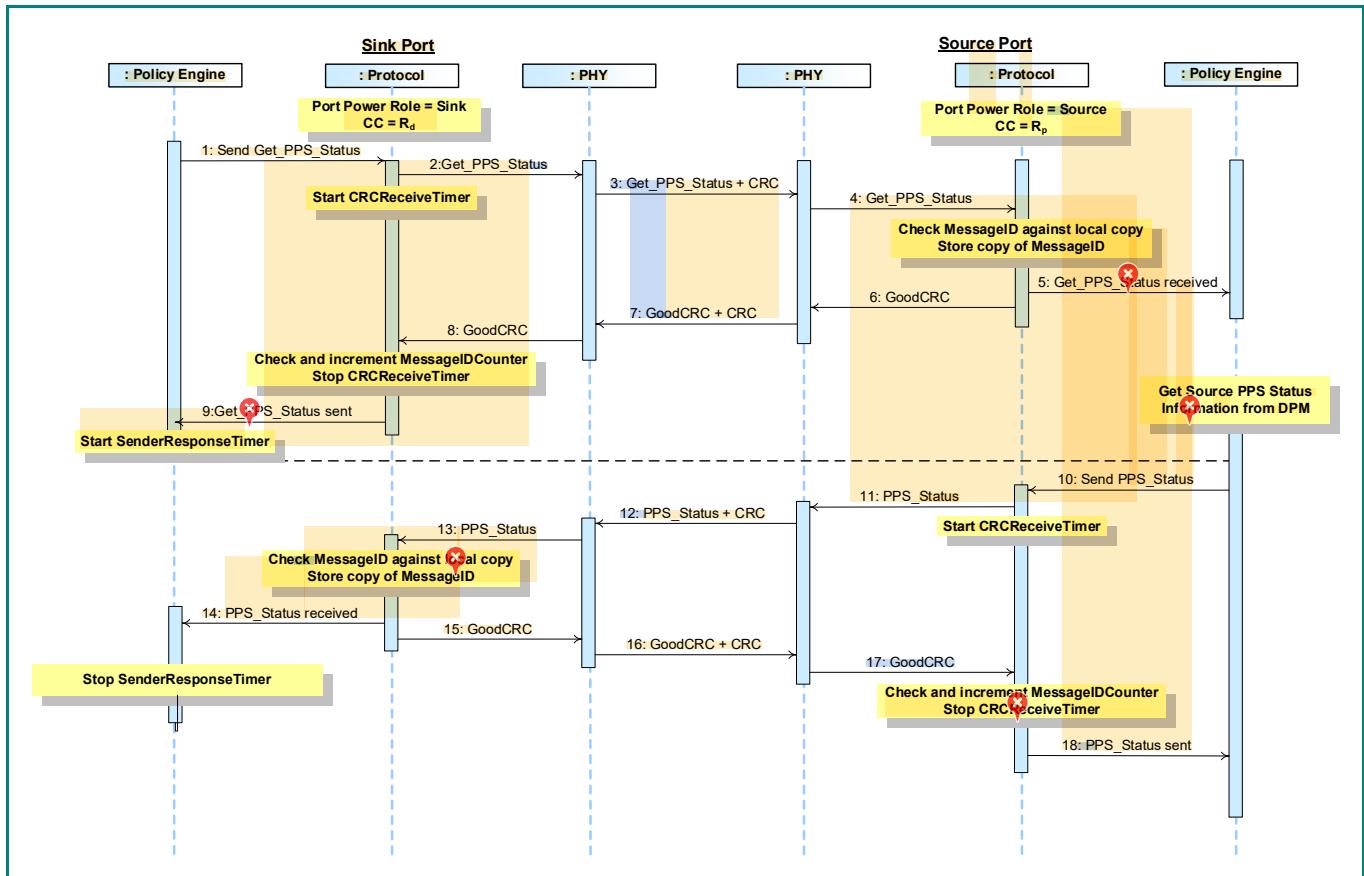
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Status</i> Message was successfully sent.
The Cable Plug has informed the VCONN Source of its present status.		

8.3.2.12.2.4

Sink Gets Source PPS Status

Figure 8-65 “Sink Gets Source PPS Status” shows an example sequence between a Source and a Sink where, after the Sink has received an alert (see [Section 8.3.2.12.2 “Status”](#)) that there has been a PPS status change, the Sink gets more details on the change.

✖ **Figure 8-65 “Sink Gets Source PPS Status”**



 **Table 8.93 "Steps for a Sink getting Source PPS status Sequence"** below provides a detailed explanation of what happens at each labeled step in **Figure 8-65 "Sink Gets Source PPS Status"** above.

Table 8.93 "Steps for a Sink getting Source PPS status Sequence"

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_PPS_Status Message.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_PPS_Status Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_PPS_Status Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Status Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_PPS_Status Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_PPS_Status Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Source status which is provided. The Policy Engine tells the Protocol Layer to form a PPS_Status Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the PPS_Status Message.	Physical Layer appends a CRC and sends the PPS_Status Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received PPS_Status Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>PPS_Status</i> Message was successfully sent.
The Source has informed the Sink of its present PPS status.		

8.3.2.12.3 Source/Sink Capabilities

8.3.2.12.3.1 SPR

8.3.2.12.3.1.1 Sink Gets Source Capabilities

Figure 8-66 “Sink Gets Source’s Capabilities” shows an example sequence between a Source and a Sink when the Sink gets the Source’s capabilities.

Figure 8-66 “Sink Gets Source’s Capabilities”

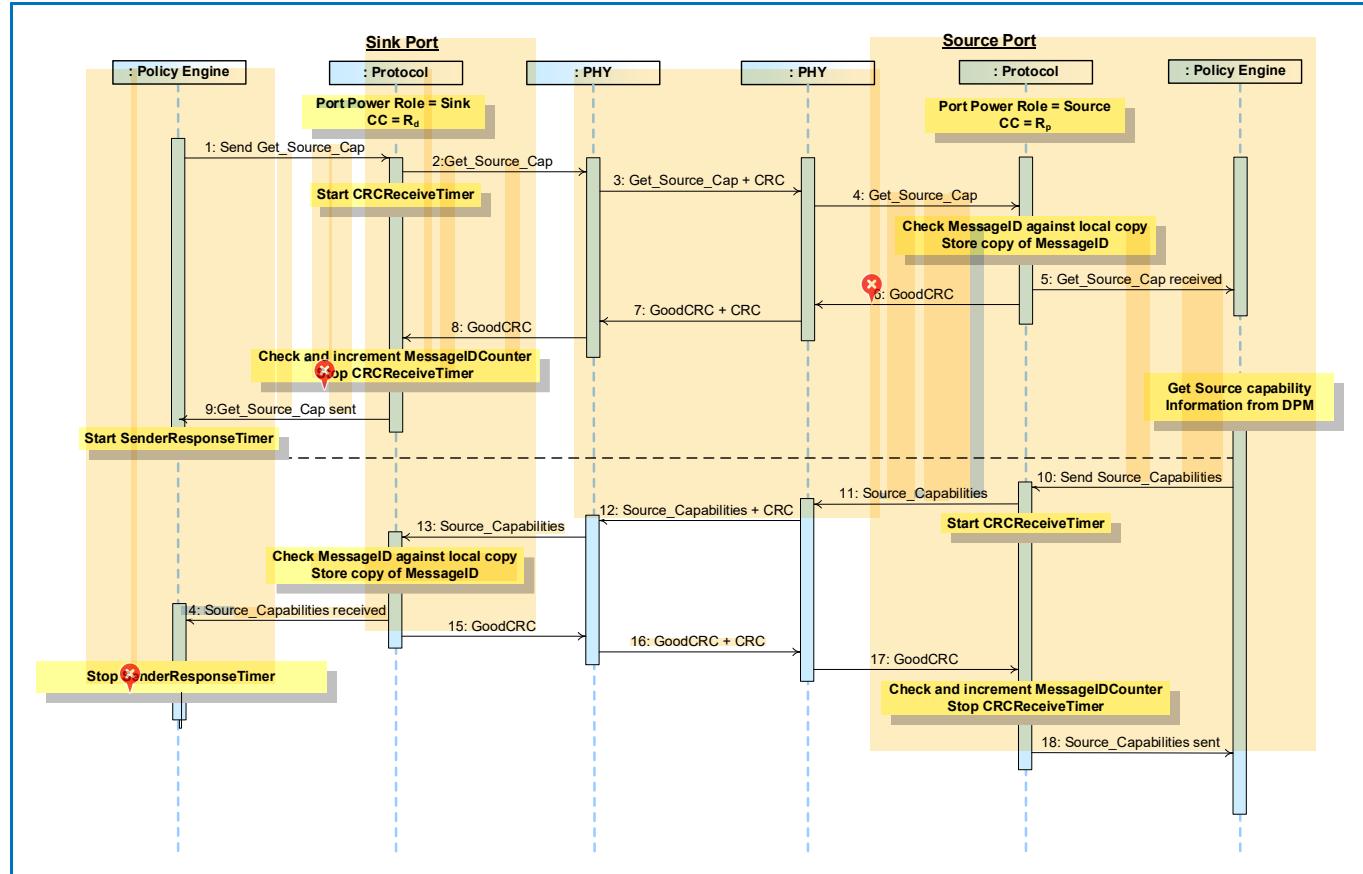


Table 8.94 “Steps for a Sink getting Source Capabilities Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-66 “Sink Gets Source’s Capabilities”** above.

Table 8.94 “Steps for a Sink getting Source Capabilities Sequence”

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Source_Cap Message.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Source_Cap Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Source_Cap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Source_Cap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Source_Cap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Source_Cap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Source capabilities which are provided. The Policy Engine tells the Protocol Layer to form a Source_Capabilities Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Source_Capabilities Message.	Physical Layer appends a CRC and sends the Source_Capabilities Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Source_Capabilities Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Capabilities</i> Message was successfully sent.
The Source has informed the Sink of its capabilities.		

8.3.2.12.3.1.2

Dual-Role Source Gets Source Capabilities from a Dual-Role Sink

Figure 8-67 “Dual-Role Source Gets Dual-Role Sink’s Capabilities as a Source” shows an example sequence between a Dual-Role Source and a Dual-Role Sink when the Source gets the Sink’s capabilities as a Source.

Figure 8-67 “Dual-Role Source Gets Dual-Role Sink’s Capabilities as a Source”

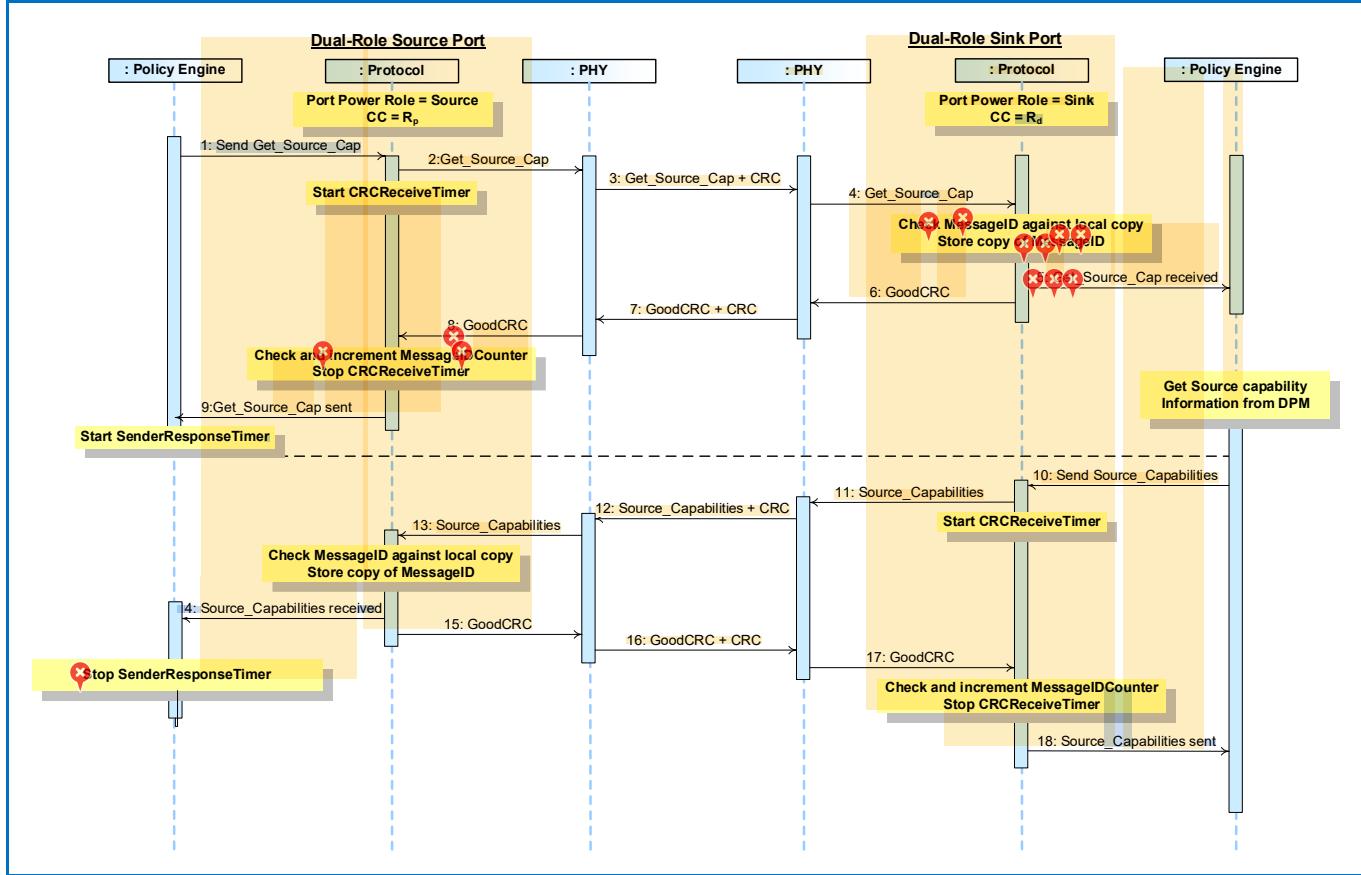


Table 8.95 “Steps for a Dual-Role Source getting Dual-Role Sink’s capabilities as a Source Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-67 “Dual-Role Source Gets Dual-Role Sink’s Capabilities as a Source”** above.

Table 8.95 “Steps for a Dual-Role Source getting Dual-Role Sink’s capabilities as a Source Sequence”

Step	Dual-Role Source Port	Dual-Role Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Source_Cap Message.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Source_Cap Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Source_Cap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Source_Cap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Source_Cap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Source_Cap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Source capabilities which are provided. The Policy Engine tells the Protocol Layer to form a Source_Capabilities Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Source_Capabilities Message.	Physical Layer appends a CRC and sends the Source_Capabilities Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Source_Capabilities Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC?</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Capabilities</i> Message was successfully sent.
The Dual-Role Sink has informed the Dual-Role Source of its capabilities.		

8.3.2.12.3.1.3

Source Gets Sink Capabilities

Figure 8-68 “Source Gets Sink’s Capabilities” shows an example sequence between a Source and a Sink when the Source gets the Sink’s capabilities.

Figure 8-68 “Source Gets Sink’s Capabilities”

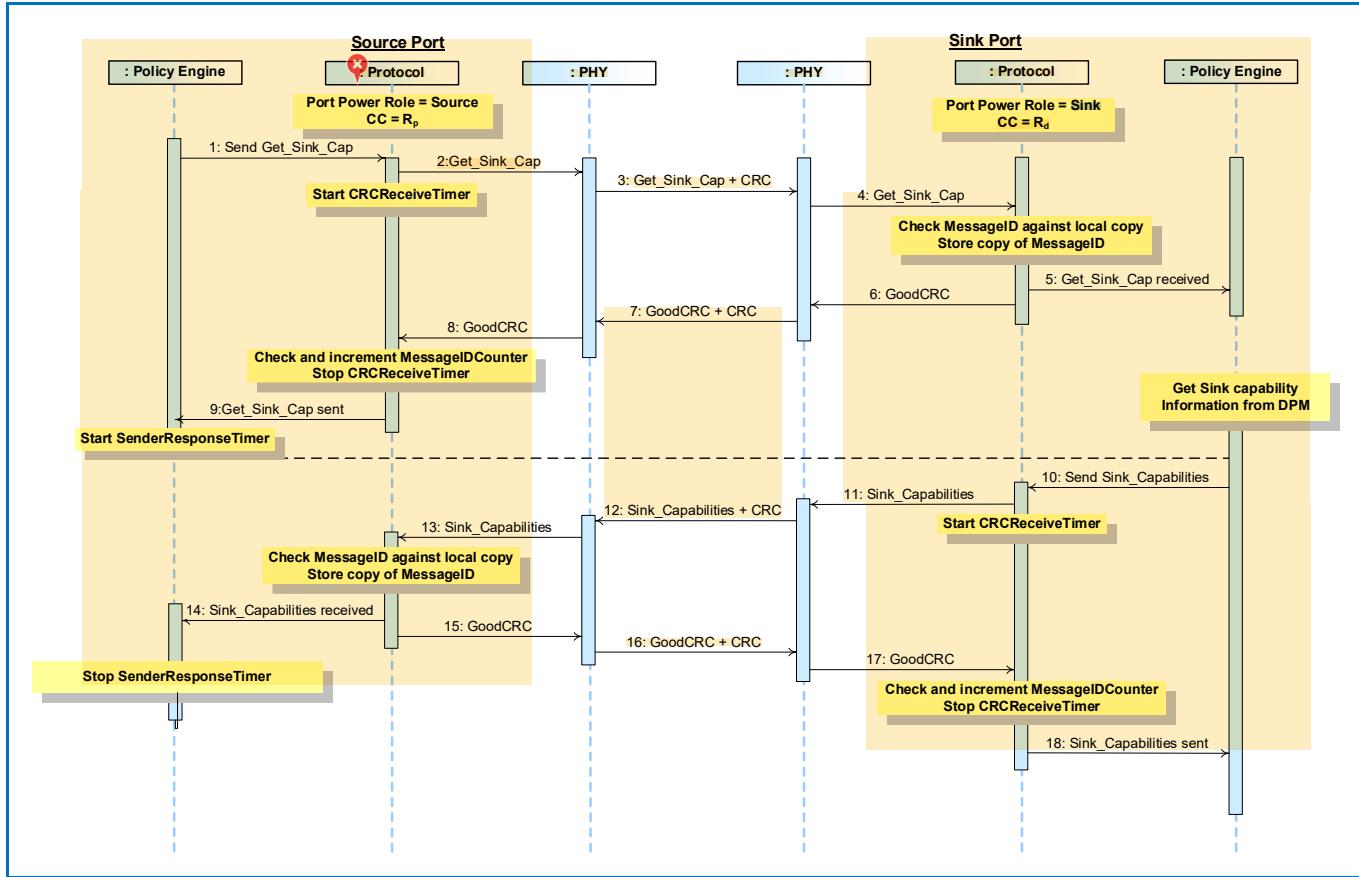


Table 8.96 “Steps for a Source getting Sink Capabilities Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-68 “Source Gets Sink’s Capabilities”** above.

Table 8.96 “Steps for a Source getting Sink Capabilities Sequence”

Step	Source Port	Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Sink_Cap Message.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Sink_Cap Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Sink_Cap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Sink_Cap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Sink_Cap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Sink_Cap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Sink capabilities which are provided. The Policy Engine tells the Protocol Layer to form a Sink_Capabilities Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Sink_Capabilities Message.	Physical Layer appends a CRC and sends the Sink_Capabilities Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Sink_Capabilities Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Sink_Capabilities</i> Message was successfully sent.
The Sink has informed the Source of its capabilities.		

8.3.2.12.3.1.4

Dual-Role Sink Get Sink Capabilities from a Dual-Role Source

Figure 8-69 “Dual-Role Sink Gets Dual-Role Source’s Capabilities as a Sink” shows an example sequence between a Dual-Role Source and a Dual-Role Sink when the Dual-Role Sink gets the Dual-Role Source’s capabilities as a Sink.

Figure 8-69 “Dual-Role Sink Gets Dual-Role Source’s Capabilities as a Sink”

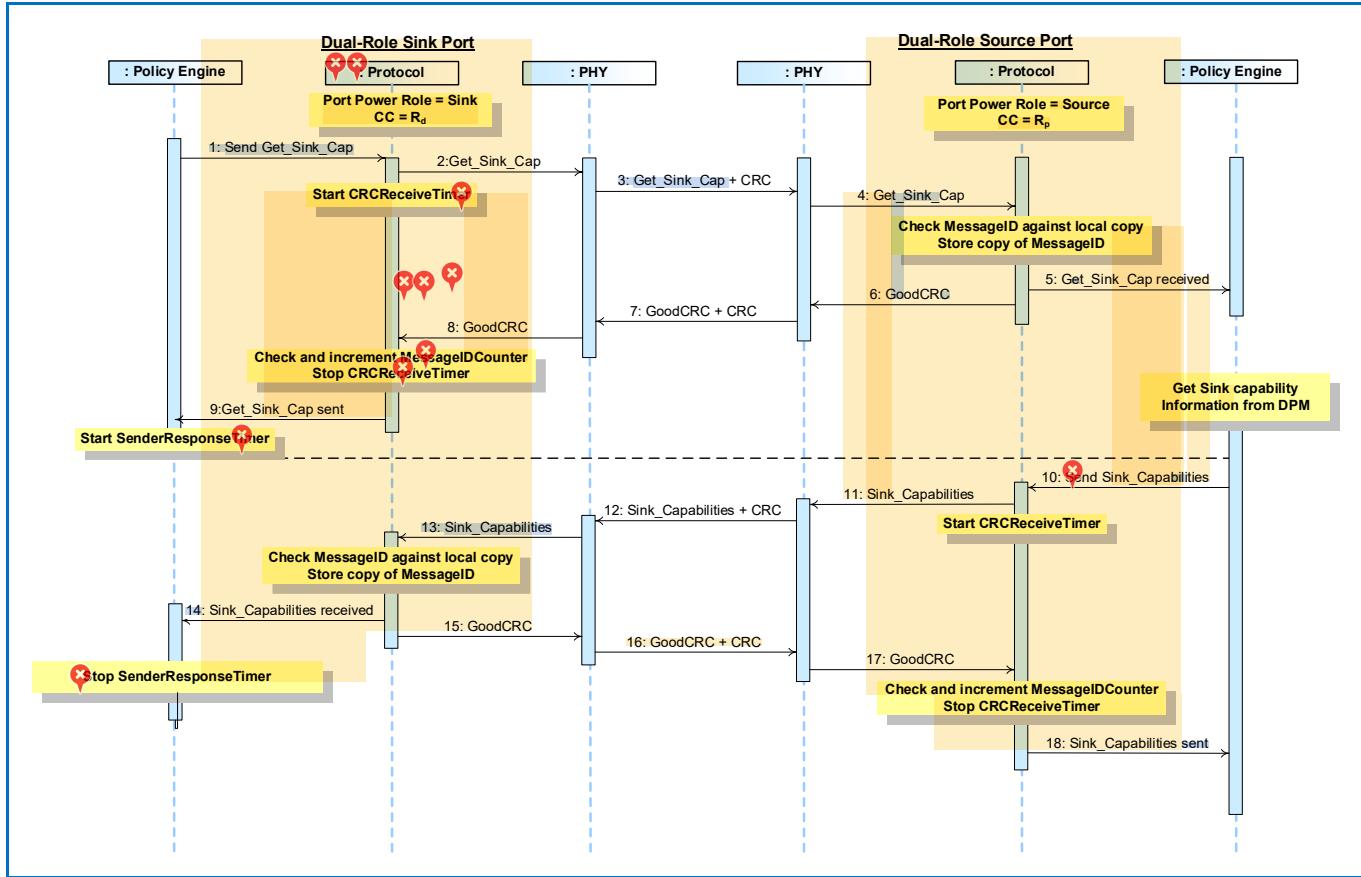


Table 8.97 "Steps for a Dual-Role Sink getting Dual-Role Source capabilities as a Sink Sequence" below provides a detailed explanation of what happens at each labeled step in **Figure 8-69 "Dual-Role Sink Gets Dual-Role Source's Capabilities as a Sink"** above.

Table 8.97 "Steps for a Dual-Role Sink getting Dual-Role Source capabilities as a Sink Sequence"

Step	Dual-Role Sink Port	Dual-Role Source Port
1	The Port has Port Power Role set to Dual-Role Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Sink_Cap Message.	The Port has Port Power Role set to Dual-Role Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Sink_Cap Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Sink_Cap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Sink_Cap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Sink_Cap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Sink_Cap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Dual-Role Source capabilities which are provided. The Policy Engine tells the Protocol Layer to form a Sink_Capabilities Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Sink_Capabilities Message.	Physical Layer appends a CRC and sends the Sink_Capabilities Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Sink_Capabilities Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC?</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Sink_Capabilities</i> Message was successfully sent.
The Dual-Role Source has informed the Dual-Role Sink of its capabilities as a Sink.		

8.3.2.12.3.2 EPR

8.3.2.12.3.2.1 Sink Gets EPR Source Capabilities

Figure 8-70 “Sink Gets Source’s EPR Capabilities” shows an example sequence between a Source and a Sink when the Sink gets the Source’s EPR capabilities.

Figure 8-70 “Sink Gets Source’s EPR Capabilities”

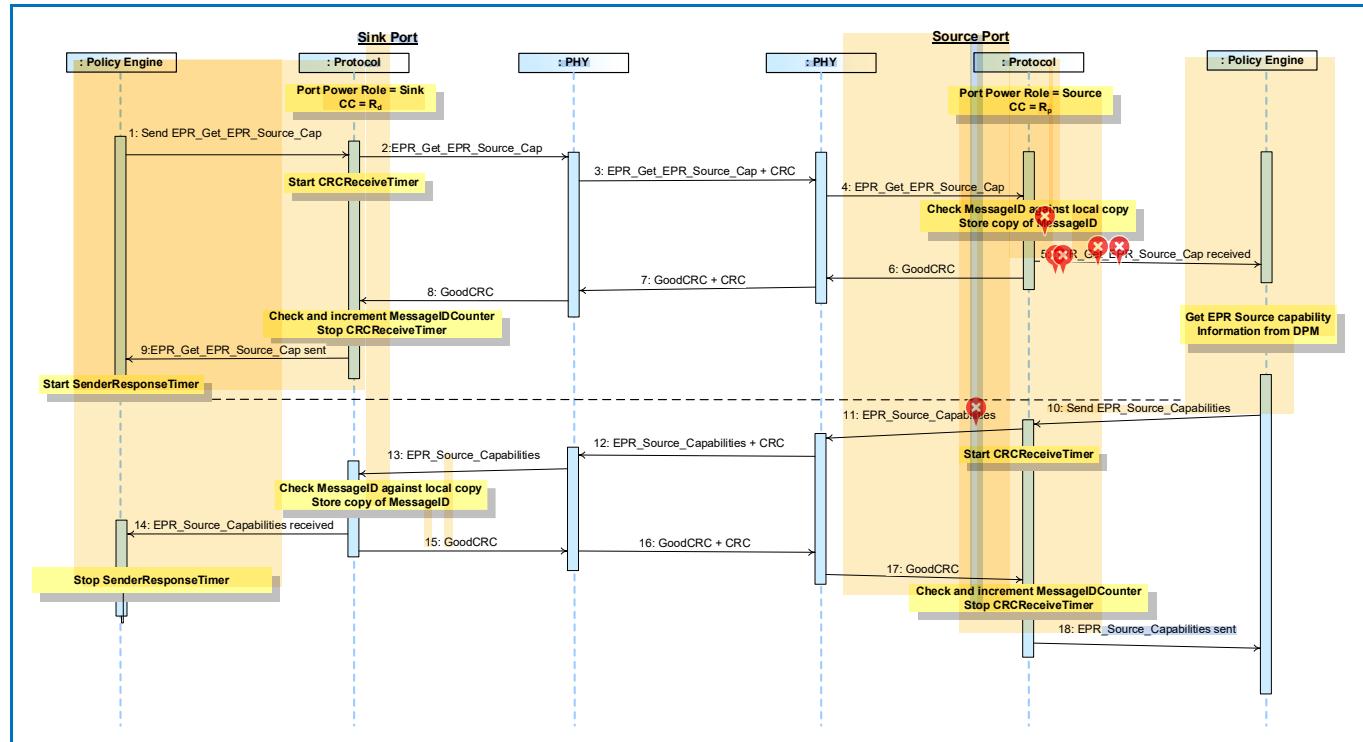


Table 8.98 “Steps for a Sink getting EPR Source Capabilities Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-70 “Sink Gets Source’s EPR Capabilities”** above.

Table 8.98 “Steps for a Sink getting EPR Source Capabilities Sequence”

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a EPR_Get_Source_Cap Message.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the EPR_Get_Source_Cap Message. Starts CRCReceiveTimer .	Physical Layer receives the EPR_Get_Source_Cap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the EPR_Get_Source_Cap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Get_Source_Cap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the EPR_Get_Source_Cap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present EPR Source capabilities which are provided. The Policy Engine tells the Protocol Layer to form an EPR_Source_Capabilities Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the EPR_Source_Capabilities Message.	Physical Layer appends a CRC and sends the EPR_Source_Capabilities Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Source_Capabilities Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Capabilities</i> Message was successfully sent.
The Source has informed the Sink of its EPR capabilities.		

8.3.2.12.3.2.2

Dual-Role Source Gets Source Capabilities from a Dual-Role EPR Sink

Figure 8-71 “Dual-Role Source Gets Dual-Role Sink’s Capabilities as an EPR Source” shows an example sequence between a Dual-Role Source and a Dual-Role Sink when the Source gets the Sink’s capabilities as an EPR Source.

Figure 8-71 “Dual-Role Source Gets Dual-Role Sink’s Capabilities as an EPR Source”

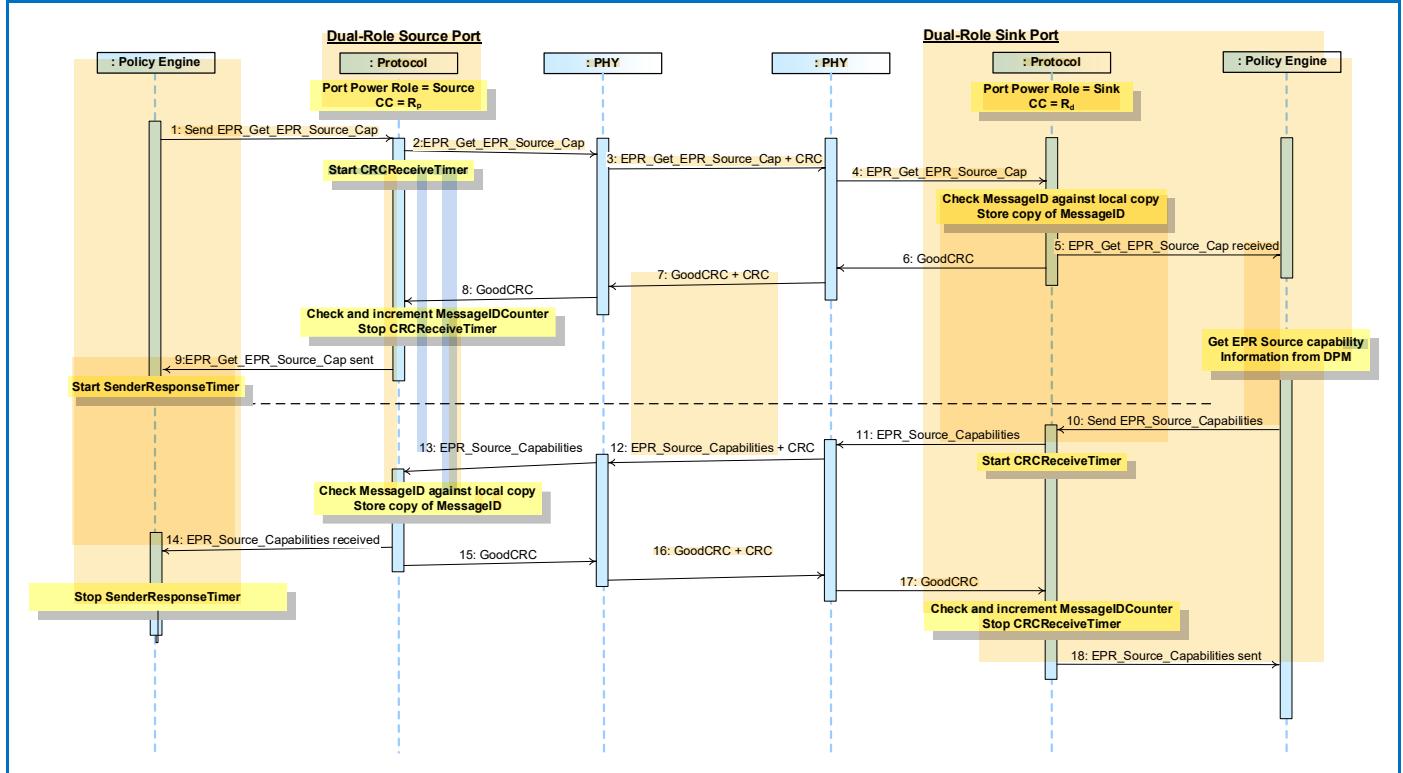


Table 8.99 “Steps for a Dual-Role Source getting Dual-Role Sink’s capabilities as an EPR Source Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-71 “Dual-Role Source Gets Dual-Role Sink’s Capabilities as an EPR Source”** above.

Table 8.99 “Steps for a Dual-Role Source getting Dual-Role Sink’s capabilities as an EPR Source Sequence”

Step	Dual-Role Source Port	Dual-Role Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a EPR_Get_Source_Cap Message.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the EPR_Get_Source_Cap Message. Starts CRCReceiveTimer .	Physical Layer receives the EPR_Get_Source_Cap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the EPR_Get_Source_Cap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Get_Source_Cap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the EPR_Get_Source_Cap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Source capabilities which are provided. The Policy Engine tells the Protocol Layer to form an EPR_Source_Capabilities Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the EPR_Source_Capabilities Message.	Physical Layer appends a CRC and sends the EPR_Source_Capabilities Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Source_Capabilities Message information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>EPR_Source_Capabilities</i> Message was successfully sent.
The Dual-Role Sink has informed the Dual-Role Source of its EPR capabilities.		

8.3.2.12.3.2.3

Source Gets Sink EPR Capabilities

Figure 8-72 “Source Gets Sink’s EPR Capabilities shows an example sequence between a Source and a Sink when the Source gets the Sink’s EPR capabilities.

Figure 8-72 “Source Gets Sink’s EPR Capabilities”

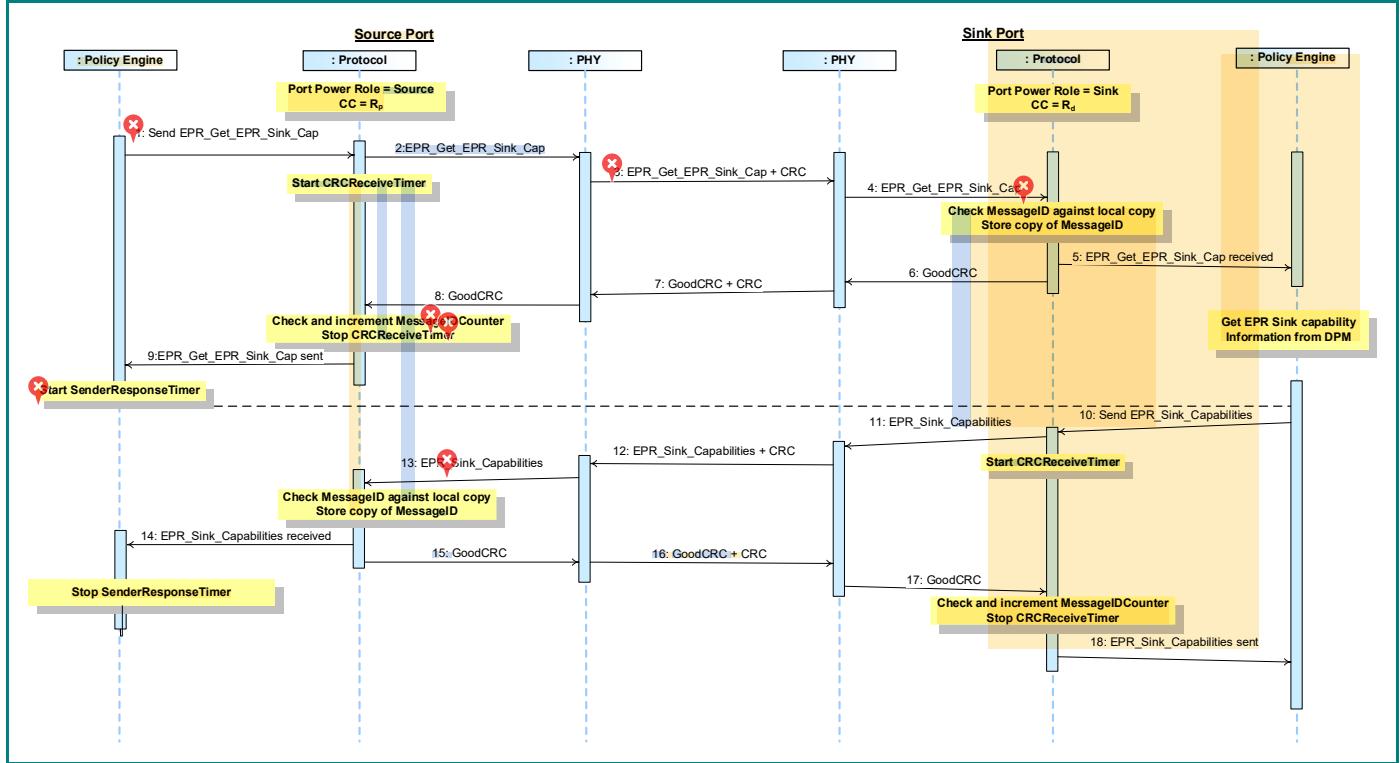


Table 8.100 “Steps for a Source getting Sink EPR Capabilities Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-72 “Source Gets Sink’s EPR Capabilities”** above.

Table 8.100 “Steps for a Source getting Sink EPR Capabilities Sequence”

Step	Source Port	Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a EPR_Get_Sink_Cap Message.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the EPR_Get_Sink_Cap Message. Starts CRCReceiveTimer .	Physical Layer receives the EPR_Get_Sink_Cap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the EPR_Get_Sink_Cap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Get_Sink_Cap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the EPR_Get_Sink_Cap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Sink capabilities which are provided. The Policy Engine tells the Protocol Layer to form an EPR_Sink_Capabilities Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the EPR_Sink_Capabilities Message.	Physical Layer appends a CRC and sends the EPR_Sink_Capabilities Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Sink_Capabilities Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
18		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the EPR_Sink_Capabilities Message was successfully sent.
The Sink has informed the Source of its EPR capabilities.		

8.3.2.12.3.2.4

Dual-Role Sink Get Sink EPR Capabilities from a Dual-Role Source

Figure 8-73 “Dual-Role Sink Gets Dual-Role Source’s Capabilities as an EPR Sink” shows an example sequence between a Dual-Role Source and a Dual-Role Sink when the Dual-Role Sink gets the Dual-Role Source’s capabilities as a Sink.

Figure 8-73 “Dual-Role Sink Gets Dual-Role Source’s Capabilities as an EPR Sink”

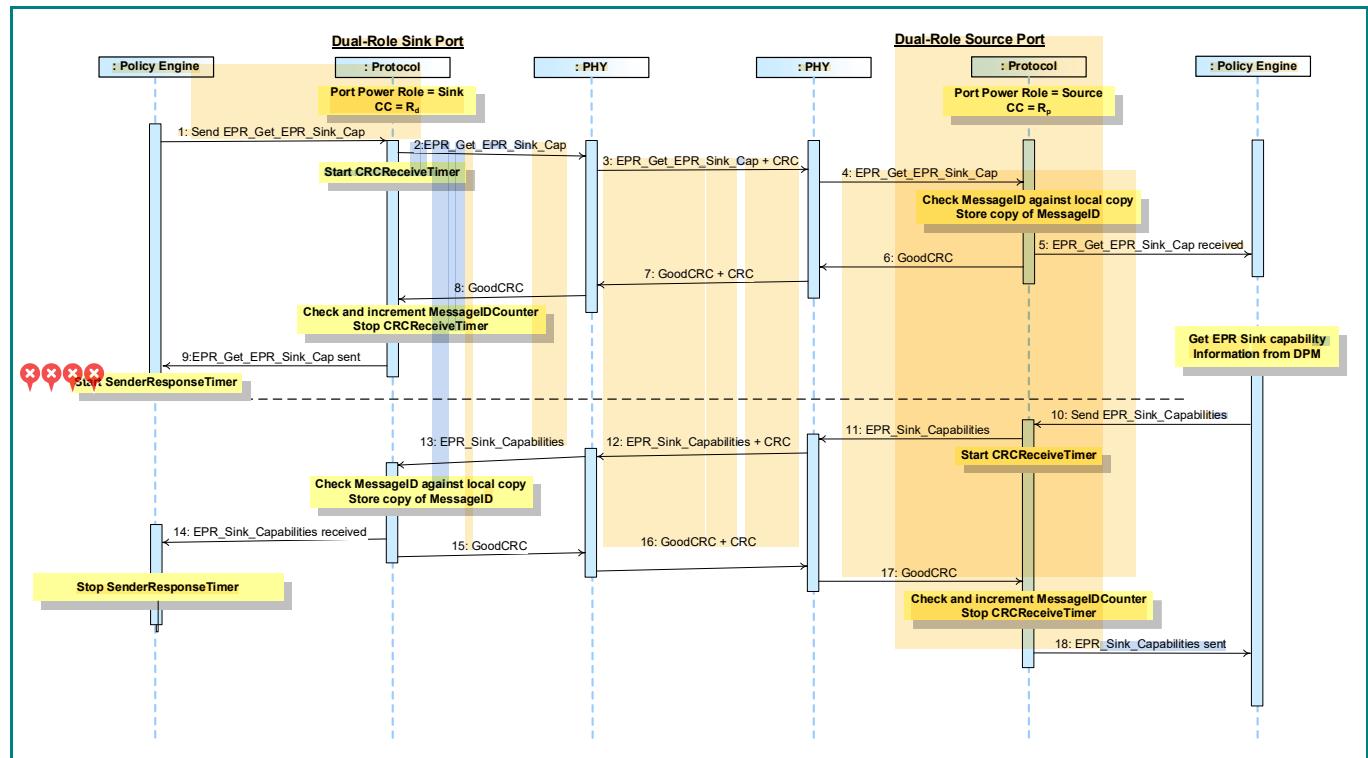


Table 8.101 “Steps for a Dual-Role Sink getting Dual-Role Source capabilities as an EPR Sink Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-73 “Dual-Role Sink Gets Dual-Role Source’s Capabilities as an EPR Sink”** above.

Table 8.101 “Steps for a Dual-Role Sink getting Dual-Role Source capabilities as an EPR Sink Sequence”

Step	Dual-Role Sink Port	Dual-Role Source Port
1	The Port has Port Power Role set to Dual-Role Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a EPR_Get_Sink_Cap Message.	The Port has Port Power Role set to Dual-Role Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the EPR_Get_Sink_Cap Message. Starts CRCReceiveTimer .	Physical Layer receives the EPR_Get_Sink_Cap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the EPR_Get_Sink_Cap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Get_Sink_Cap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the EPR_Get_Sink_Cap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Dual-Role Source capabilities which are provided. The Policy Engine tells the Protocol Layer to form an EPR_Sink_Capabilities Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the EPR_Sink_Capabilities Message.	Physical Layer appends a CRC and sends the EPR_Sink_Capabilities Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received EPR_Sink_Capabilities Message information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>EPR_Sink_Capabilities</i> Message was successfully sent.
The Dual-Role Source has informed the Dual-Role Sink of its capabilities as an EPR Sink.		

8.3.2.12.4

Extended Capabilities

8.3.2.12.4.1

Sink Gets Source Extended Capabilities

Figure 8-74 “Sink Gets Source’s Extended Capabilities” shows an example sequence between a Source and a Sink when the Sink gets the Source’s extended capabilities.

Figure 8-74 “Sink Gets Source’s Extended Capabilities”

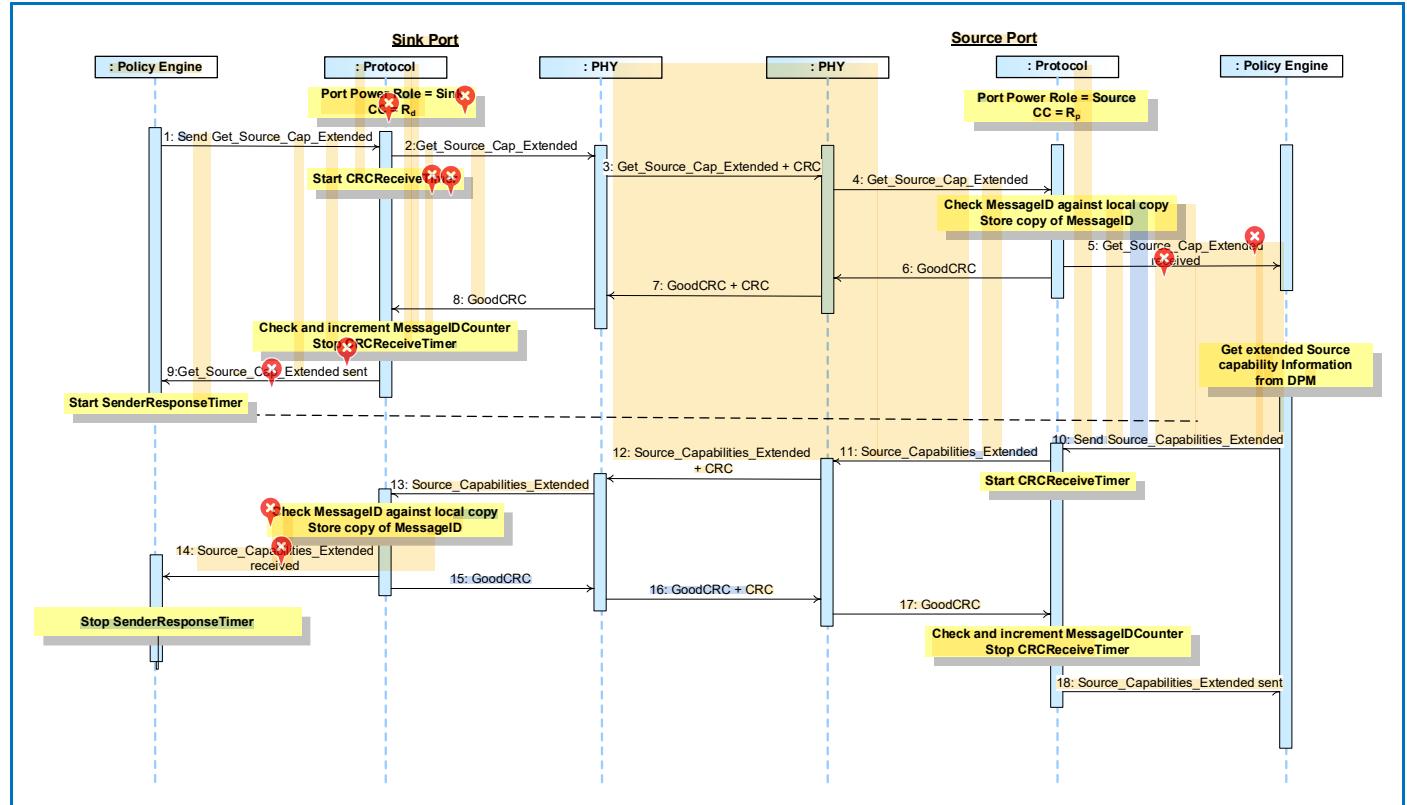


Table 8.102 “Steps for a Sink getting Source extended capabilities Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-74 “Sink Gets Source’s Extended Capabilities”** above.

Table 8.102 “Steps for a Sink getting Source extended capabilities Sequence”

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Source_Cap_Extended Message.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Source_Cap_Extended Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Source_Cap_Extended Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Source_Cap_Extended Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Source_Cap_Extended Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Source_Cap_Extended Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present extended Source capabilities which are provided. The Policy Engine tells the Protocol Layer to form a Source_Capabilities_Extended Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Source_Capabilities_Extended Message.	Physical Layer appends a CRC and sends the Source_Capabilities_Extended Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Source_Capabilities_Extended Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Capabilities_Extended</i> Message was successfully sent.
The Source has informed the Sink of its extended capabilities.		

8.3.2.12.4.2

Dual-Role Source Gets Source Capabilities Extended from a Dual-Role Sink

Figure 8-75 “Dual-Role Source Gets Dual-Role Sink’s Extended Capabilities” shows an example sequence between a Source and a Sink when the Dual-Role Source gets the Dual-Role Sink’s extended capabilities as a Source.

Figure 8-75 “Dual-Role Source Gets Dual-Role Sink’s Extended Capabilities”

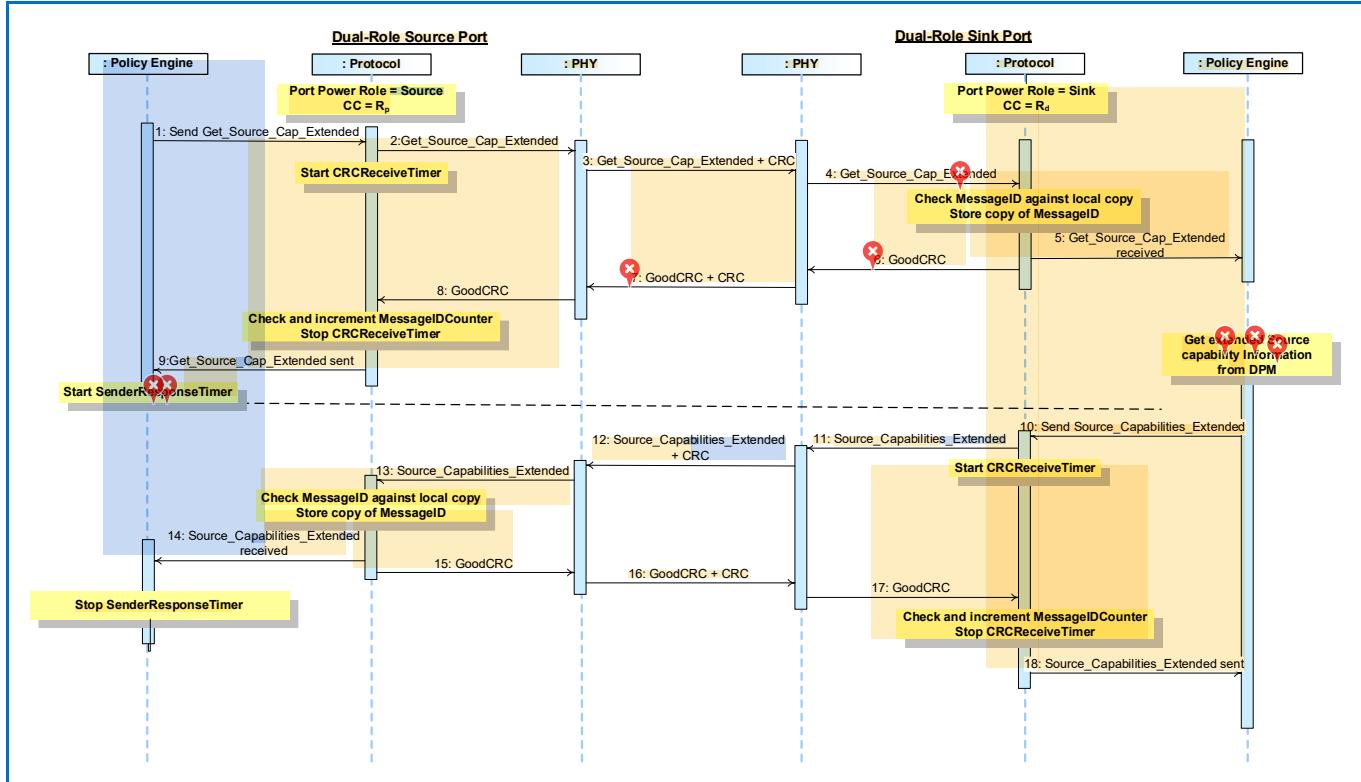


Table 8.103 “Steps for a Dual-Role Source getting Dual-Role Sink extended capabilities Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-75 “Dual-Role Source Gets Dual-Role Sink’s Extended Capabilities”** above.

Table 8.103 “Steps for a Dual-Role Source getting Dual-Role Sink extended capabilities Sequence”

Step	Dual-Role Source Port	Dual-Role Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Source_Cap_Extended Message.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Source_Cap_Extended Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Source_Cap_Extended Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Source_Cap_Extended Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Source_Cap_Extended Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Source_Cap_Extended Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present extended Source capabilities which are provided. The Policy Engine tells the Protocol Layer to form a Source_Capabilities_Extended Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Source_Capabilities_Extended Message.	Physical Layer appends a CRC and sends the Source_Capabilities_Extended Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Source_Capabilities_Extended Message information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Capabilities_Extended</i> Message was successfully sent.
The Dual-Role Sink has informed the Dual-Role Source of its extended capabilities as a Source.		

8.3.2.12.4.3

Source Gets Sink Extended Capabilities

Figure 8-76 “Source Gets Sink’s Extended Capabilities” shows an example sequence between a Source and a Sink when the Source gets the Sink’s extended capabilities.

Figure 8-76 “Source Gets Sink’s Extended Capabilities”

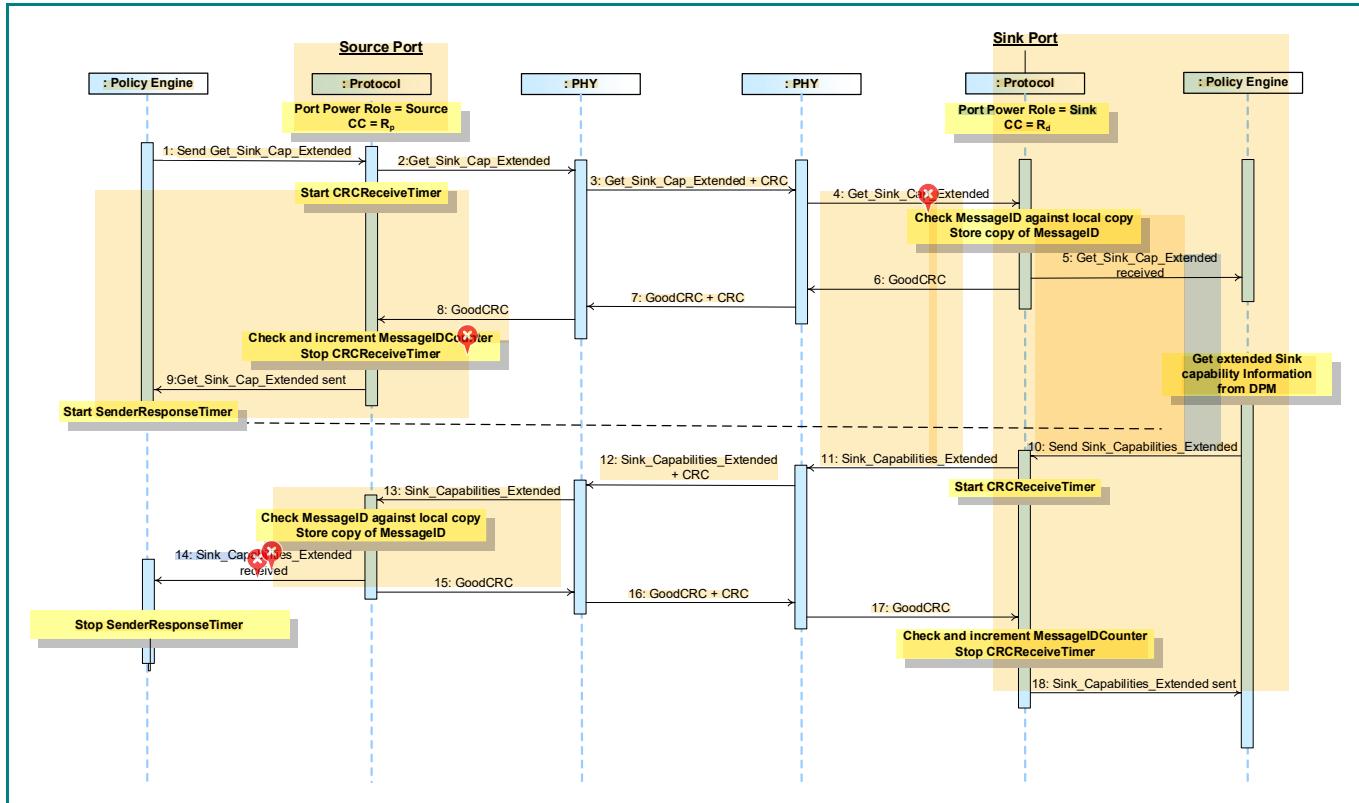


Table 8.104 “Steps for a Source getting Sink extended capabilities Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-76 “Source Gets Sink’s Extended Capabilities”** above.

Table 8.104 “Steps for a Source getting Sink extended capabilities Sequence”

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Sink_Cap_Extended Message.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Source_Cap_Extended Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Sink_Cap_Extended Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Sink_Cap_Extended Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Sink_Cap_Extended Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Sink_Cap_Extended Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present extended Source capabilities which are provided. The Policy Engine tells the Protocol Layer to form a Sink_Capabilities_Extended Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Sink_Capabilities_Extended Message.	Physical Layer appends a CRC and sends the Sink_Capabilities_Extended Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Sink_Capabilities_Extended Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Sink_Capabilities_Extended</i> Message was successfully sent.
The Sink has informed the Source of its extended capabilities.		

8.3.2.12.4.4

Dual-Role Sink Gets Sink Capabilities Extended from a Dual-Role Source

Figure 8-77 “Dual-Role Sink Gets Dual-Role Source’s Extended Capabilities” shows an example sequence between a Source and a Sink when the Dual-Role Sink gets the Dual-Role Source’s extended capabilities as a Sink.

Figure 8-77 “Dual-Role Sink Gets Dual-Role Source’s Extended Capabilities”

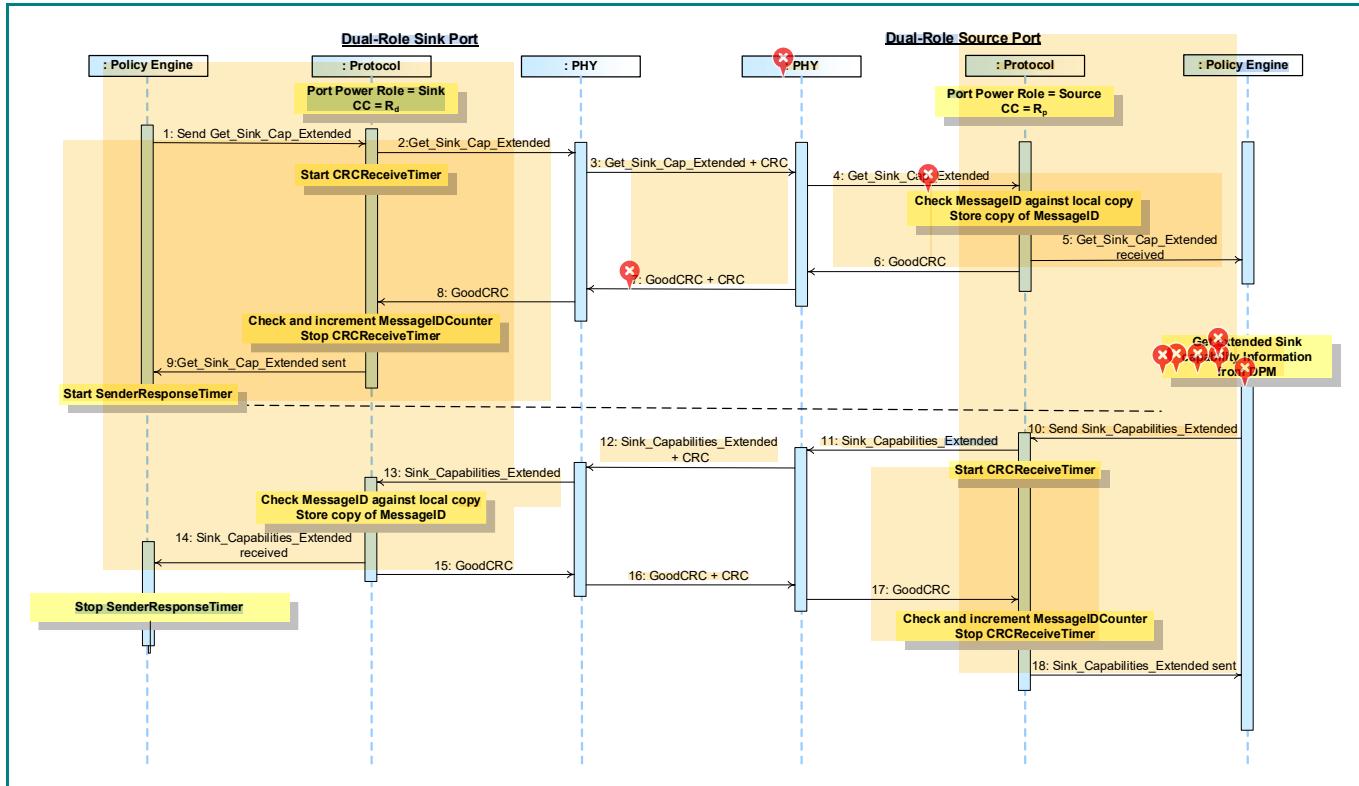


Table 8.105 “Steps for a Dual-Role Sink getting Dual-Role Source extended capabilities Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-77 “Dual-Role Sink Gets Dual-Role Source’s Extended Capabilities”** above.

Table 8.105 “Steps for a Dual-Role Sink getting Dual-Role Source extended capabilities Sequence”

Step	Dual-Role Source Port	Dual-Role Sink Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Sink_Cap_Extended Message.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Sink_Cap_Extended Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Sink_Cap_Extended Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Sink_Cap_Extended Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Sink_Cap_Extended Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Sink_Cap_Extended Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present extended Source capabilities which are provided. The Policy Engine tells the Protocol Layer to form a Sink_Capabilities_Extended Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Sink_Capabilities_Extended Message.	Physical Layer appends a CRC and sends the Sink_Capabilities_Extended Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Sink_Capabilities_Extended Message information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Sink_Capabilities_Extended</i> Message was successfully sent.
The Dual-Role Source has informed the Dual-Role Sink of its extended capabilities as a Sink.		

8.3.2.12.5 Battery Capabilities and Status

8.3.2.12.5.1 Sink Gets Battery Capabilities

Figure 8-78 “Sink Gets Source’s Battery Capabilities” shows an example sequence between a Source and a Sink when the Sink gets the Source’s Battery capabilities for a given Battery.

Figure 8-78 “Sink Gets Source’s Battery Capabilities”

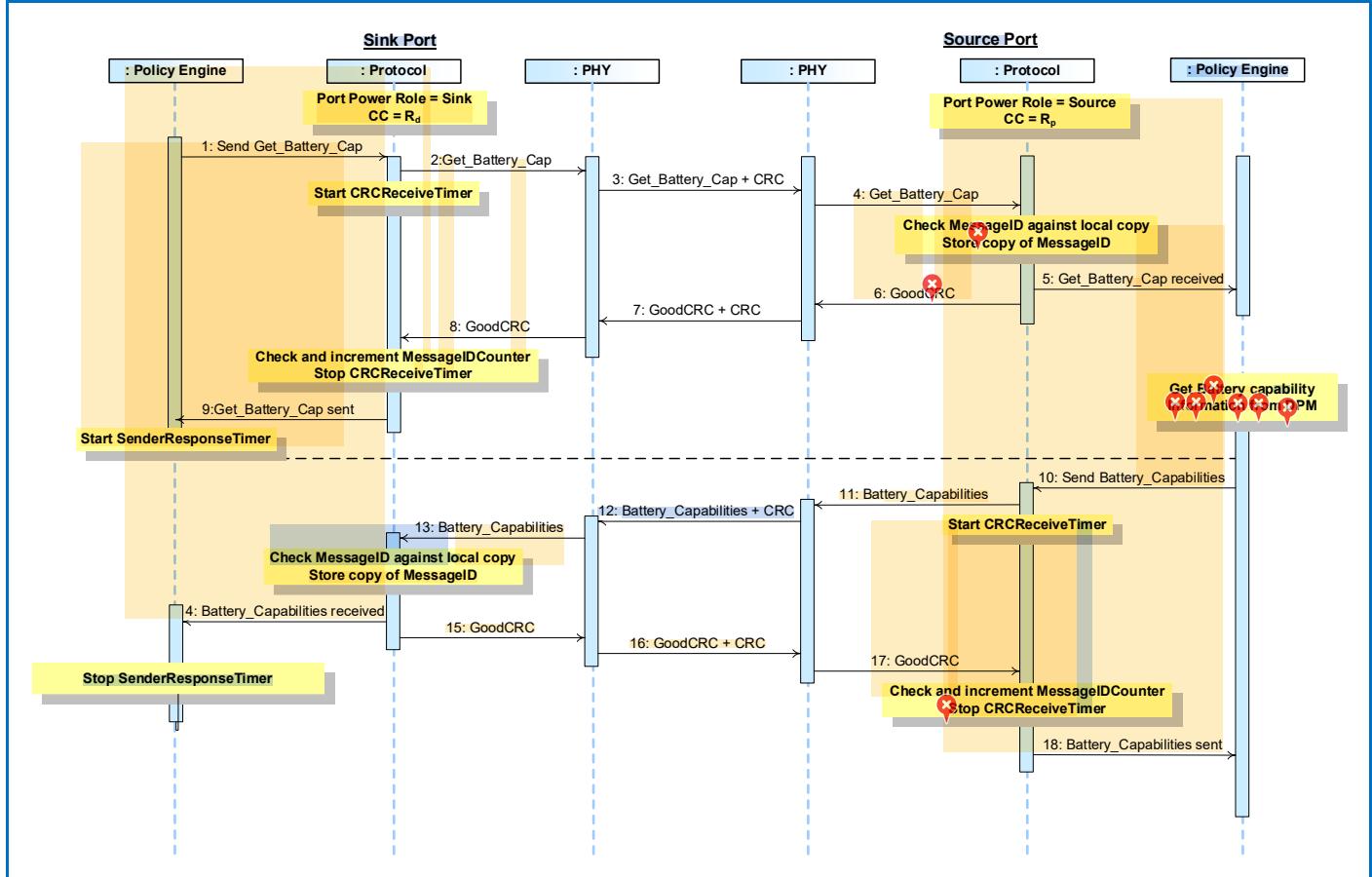


Table 8.106 “Steps for a Sink getting Source Battery capabilities Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-78 “Sink Gets Source’s Battery Capabilities”** above.

Table 8.106 “Steps for a Sink getting Source Battery capabilities Sequence”

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Battery_Cap Message containing the number of the Battery for which capabilities are being requested.📍	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Battery_Cap Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Battery_Cap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Battery_Cap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Battery_Cap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Battery_Cap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Source Battery capabilities, for the requested Battery number, which are provided. The Policy Engine tells the Protocol Layer to form a Battery_Capabilities Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Battery_Capabilities Message.	Physical Layer appends a CRC and sends the Battery_Capabilities Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Battery_Capabilities Message information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Battery_Capabilities</i> Message was successfully sent.
The Source has informed the Sink of the Battery capabilities for the requested Battery.		

8.3.2.12.5.2

Source Gets Battery Capabilities

Figure 8-79 “Source Gets Sink’s Battery Capabilities” shows an example sequence between a Source and a Sink when the Source gets the Sink’s Battery capabilities for a given Battery.

Figure 8-79 “Source Gets Sink’s Battery Capabilities”

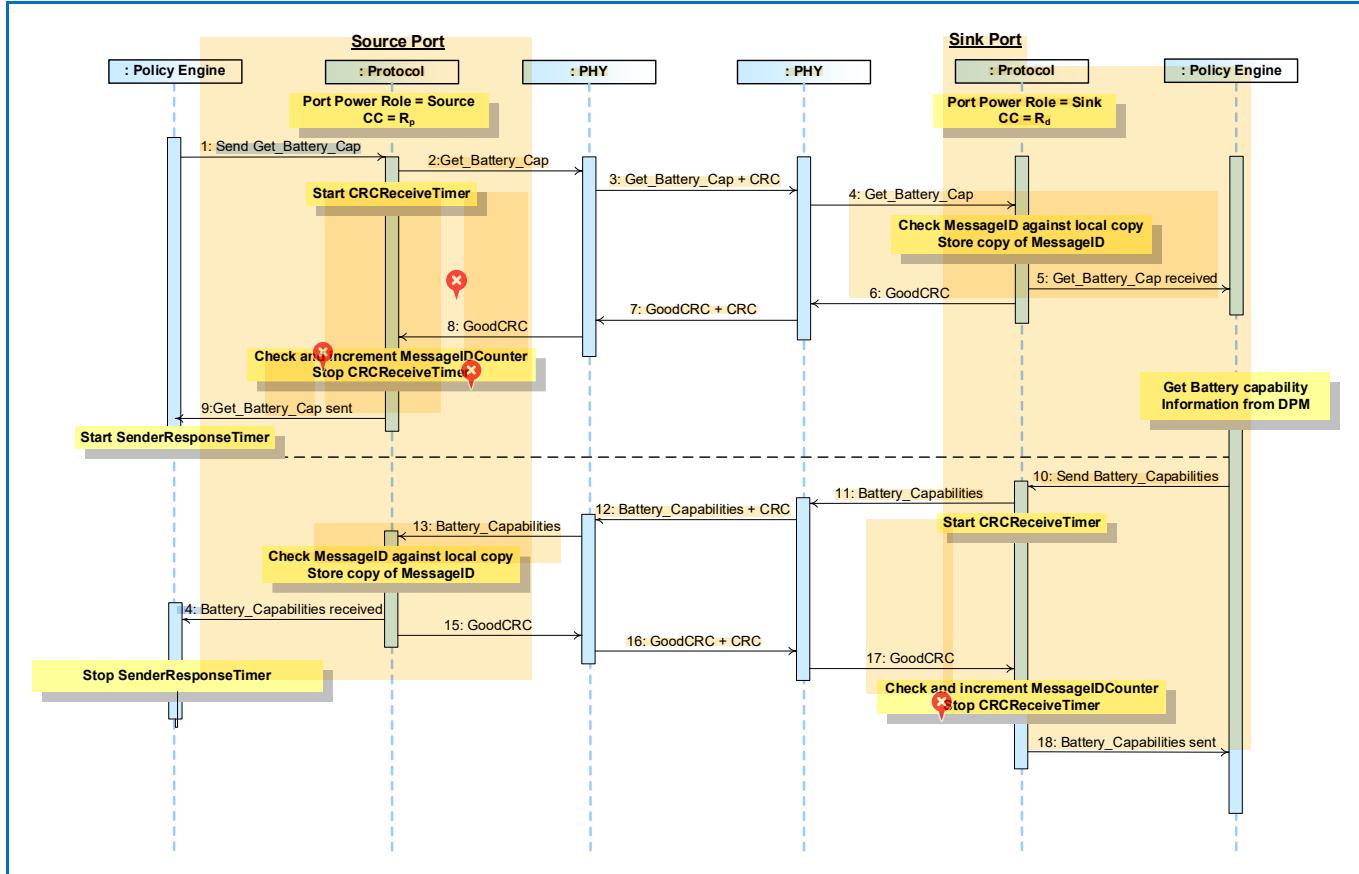


Table 8-44 below provides a detailed explanation of what happens at each labeled step in [Figure 8-79 “Source Gets Sink’s Battery Capabilities”](#) above.

Table 8.107 “Steps for a Source getting Sink Battery capabilities Sequence”

Step	Source Port	Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Battery_Cap Message containing the number of the Battery for which capabilities are being requested. 	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the  Get_Battery_Cap Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Battery_Cap Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Battery_Cap Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Battery_Cap Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Battery_Cap Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Source Battery capabilities, for the requested Battery number, which are provided. The Policy Engine tells the Protocol Layer to form a Battery_Capabilities Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Battery_Capabilities Message.	Physical Layer appends a CRC and sends the Battery_Capabilities Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Battery_Capabilities Message information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Battery_Capabilities</i> Message was successfully sent.
The Sink has informed the Source of the Battery capabilities for the requested Battery.		

8.3.2.12.5.3

Sink Gets Battery Status

Figure 8-80 “Sink Gets Source’s Battery Status” shows an example sequence between a Source and a Sink when the Sink gets the Source’s Battery status for a given Battery.

Figure 8-80 “Sink Gets Source’s Battery Status”

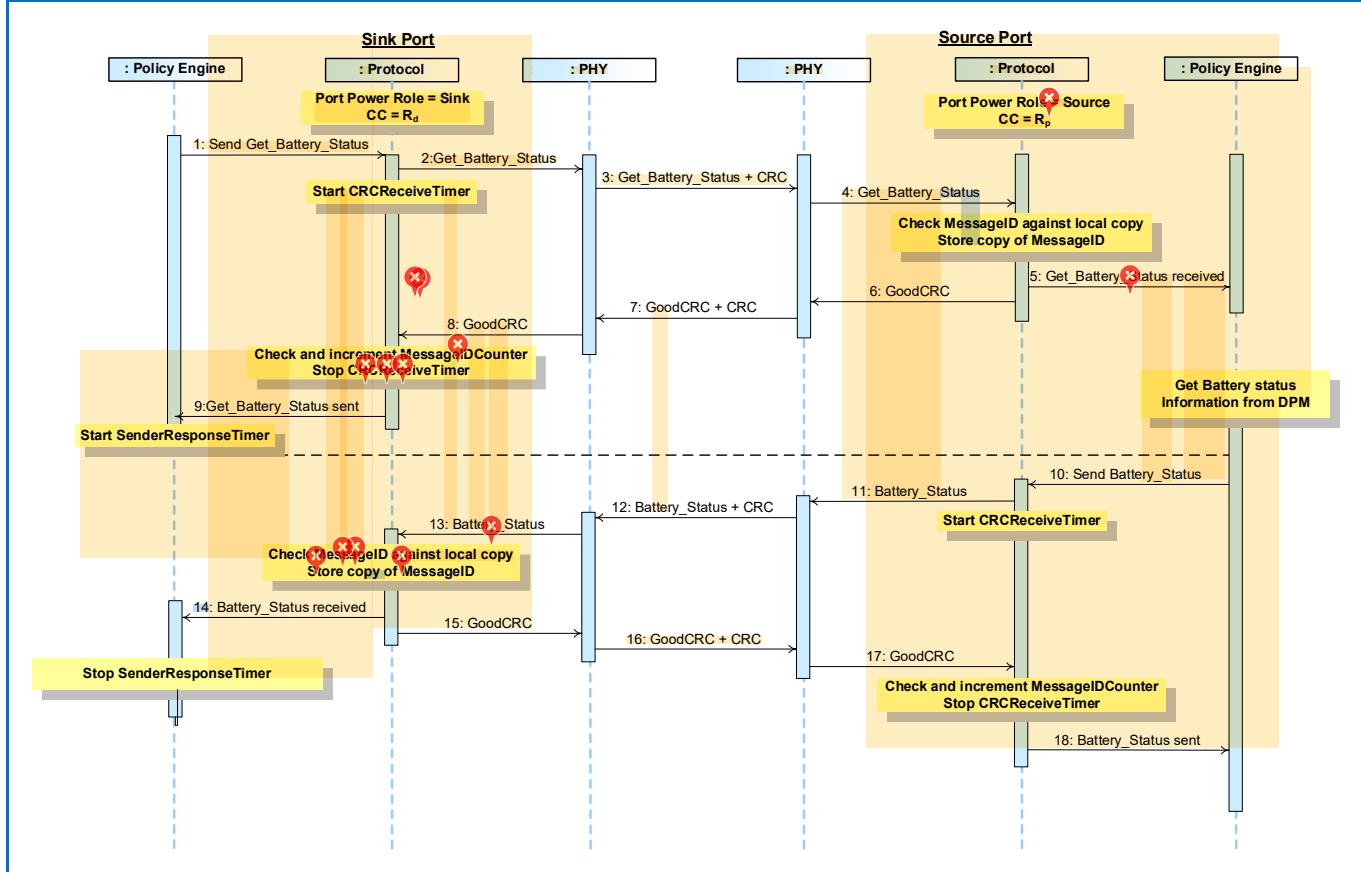


Table 8.108 “Steps for a Sink getting Source Battery status Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-80 “Sink Gets Source’s Battery Status”** above.

Table 8.108 “Steps for a Sink getting Source Battery status Sequence”

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Battery_Status Message containing the number of the Battery for which status is being requested.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Battery_Status Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Battery_Status Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Battery_Status Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Battery_Status Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Battery_Status Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Source Battery status, for the requested Battery number, which are provided. The Policy Engine tells the Protocol Layer to form a Battery_Status Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Battery_Status Message.	Physical Layer appends a CRC and sends the Battery_Status Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Battery_Status Message information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Battery_Status</i> Message was successfully sent.
The Source has informed the Sink of the Battery status for the requested Battery.		

8.3.2.12.5.4

Source Gets Battery Status

Figure 8-81 “Source Gets Sink’s Battery Status” shows an example sequence between a Source and a Sink when the Source gets the Sink’s Battery status for a given Battery.

Figure 8-81 “Source Gets Sink’s Battery Status”

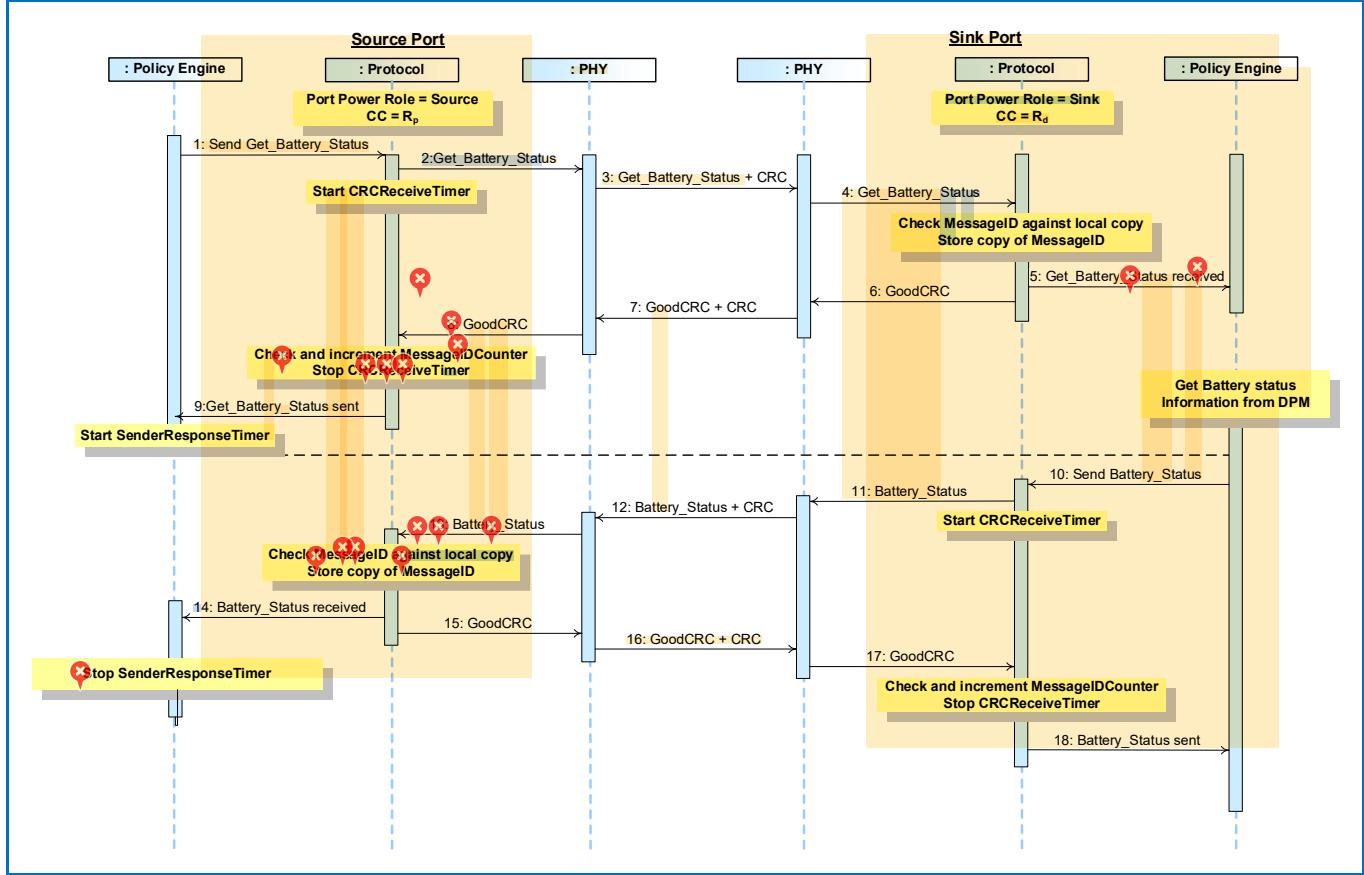


Table 8.109 “Steps for a Source getting Sink Battery status Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-81 “Source Gets Sink’s Battery Status”** above.

Table 8.109 “Steps for a Source getting Sink Battery status Sequence”

Step	Source Port	Sink Port
1	The Port has Port Power Role set to Source and the Rp pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Battery_Status Message containing the number of the Battery for which status is being requested. 	The Port has Port Power Role set to Sink with the Rd pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Battery_Status Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Battery_Status Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Battery_Status Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Battery_Status Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message. 	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Battery_Status Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Source Battery status, for the requested Battery number, which are provided. The Policy Engine tells the Protocol Layer to form a Battery_Status Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Battery_Status Message.	Physical Layer appends a CRC and sends the Battery_Status Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Battery_Status Message information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Battery_Status</i> Message was successfully sent.
The Sink has informed the Source of the Battery status for the requested Battery.		

8.3.2.12.6

Manufacturer Information

8.3.2.12.6.1

Source Gets Port Manufacturer Information from a Sink

Figure 8-82 “Source Gets Sink’s Port Manufacturer Information” shows an example sequence between a Source and a Sink when the Source gets the Sink’s Manufacturer information for the Port.

Figure 8-82 “Source Gets Sink’s Port Manufacturer Information”

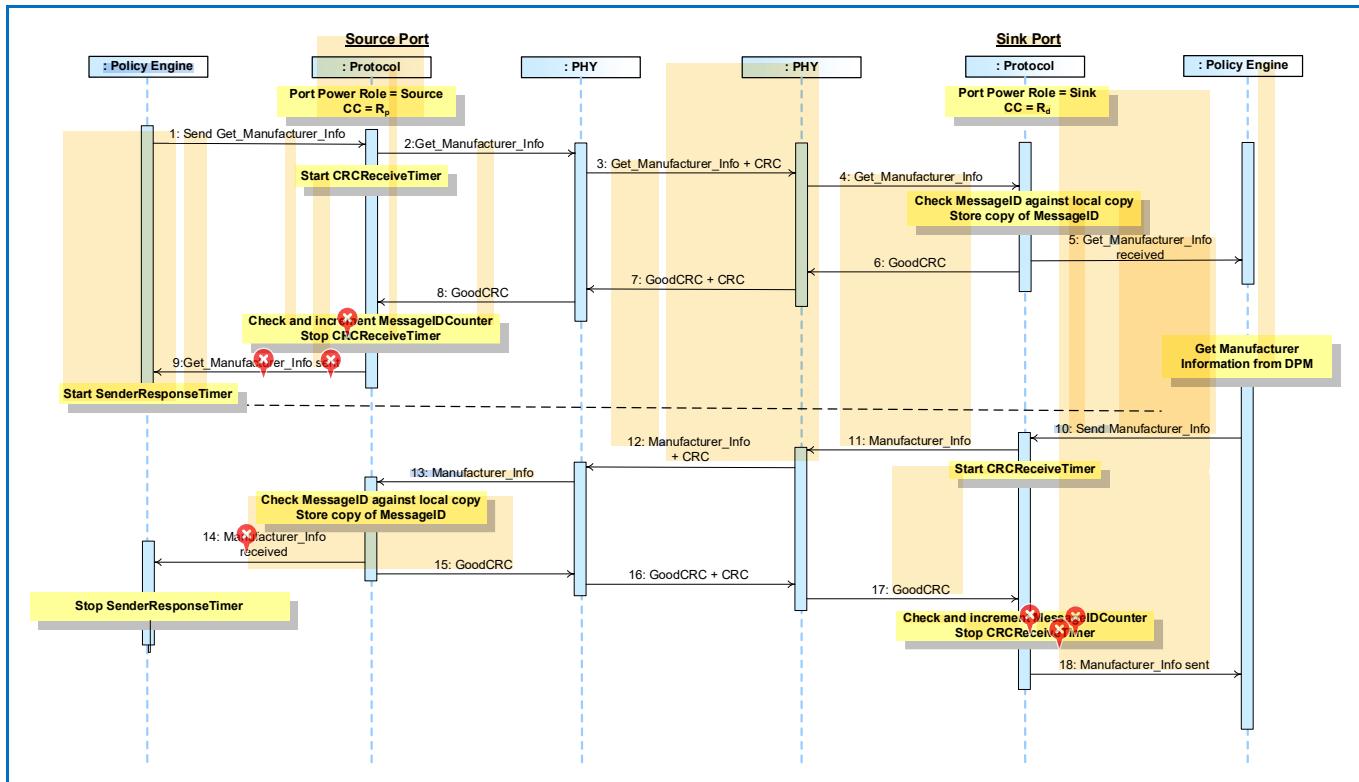


Table 8.110 “Steps for a Source getting Sink’s Port Manufacturer Information Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-82 “Source Gets Sink’s Port Manufacturer Information”** above.

Table 8.110 “Steps for a Source getting Sink’s Port Manufacturer Information Sequence”

Step	Source Port	Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Manufacturer_Info Message with a request for Port information.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Manufacturer_Info Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Manufacturer_Info Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Manufacturer_Info Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Manufacturer_Info Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Manufacturer_Info Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Port’s manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a Manufacturer_Info Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Manufacturer_Info Message.	Physical Layer appends a CRC and sends the Manufacturer_Info Message. Starts CRCReceiveTimer .

13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Manufacturer_Info</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Manufacturer_Info</i> Message was successfully sent.

The Sink has informed the Source of the manufacturer information for the Port.

8.3.2.12.6.2

Sink Gets Port Manufacturer Information from a Source

Figure 8-83 “Sink Gets Source’s Port Manufacturer Information” shows an example sequence between a Source and a Sink when the Source gets the Sink’s Manufacturer information for the Port.

Figure 8-83 “Sink Gets Source’s Port Manufacturer Information”

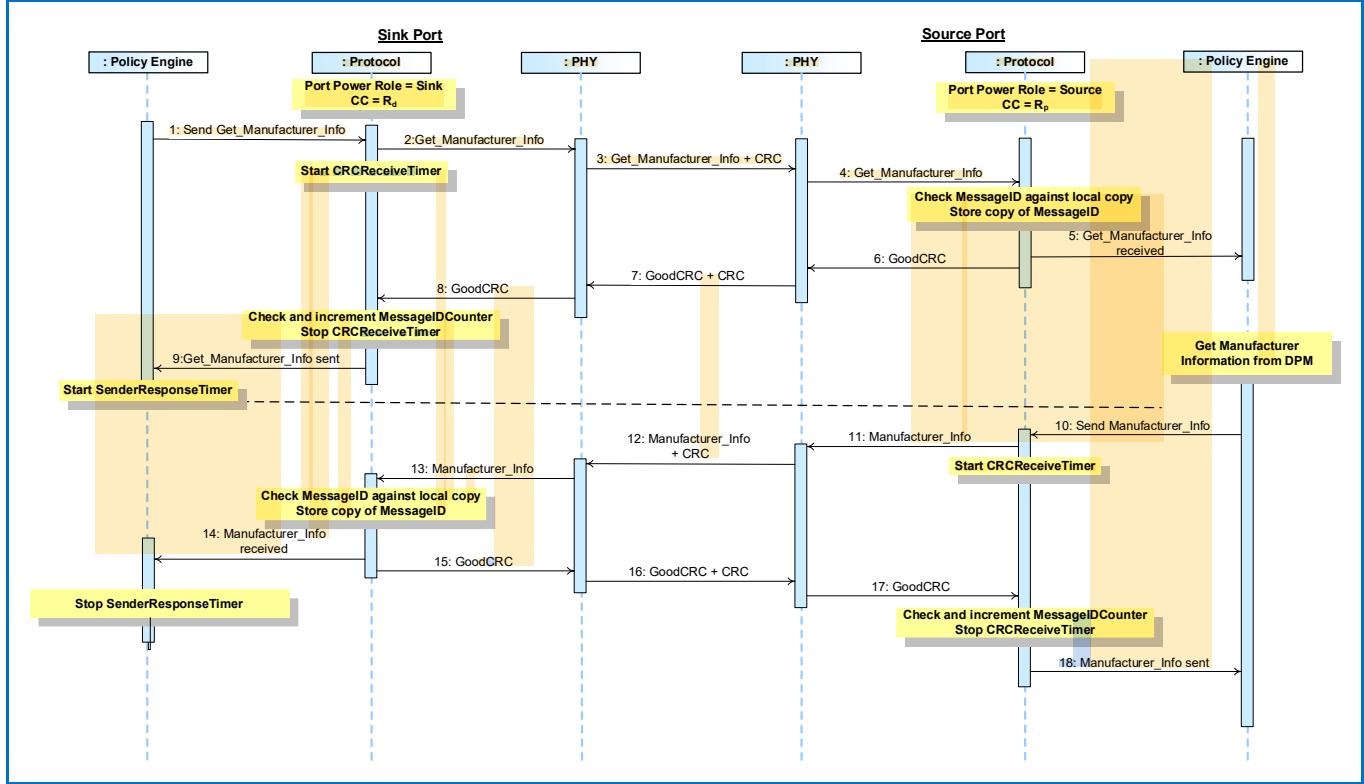


Table 8.111 “Steps for a Source getting Sink’s Port Manufacturer Information Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-83 “Sink Gets Source’s Port Manufacturer Information”** above.

Table 8.111 “Steps for a Source getting Sink’s Port Manufacturer Information Sequence”

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Manufacturer_Info Message with a request for Port information.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Manufacturer_Info Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Manufacturer_Info Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Manufacturer_Info Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Manufacturer_Info Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Manufacturer_Info Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Port’s manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a Manufacturer_Info Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Manufacturer_Info Message.	Physical Layer appends a CRC and sends the Manufacturer_Info Message. Starts CRCReceiveTimer .

13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Manufacturer_Info</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Manufacturer_Info</i> Message was successfully sent.

The Sink has informed the Source of the manufacturer information for the Port.

8.3.2.12.6.3

Source Gets Battery Manufacturer Information from a Sink

Figure 8-84 “Source Gets Sink’s Battery Manufacturer Information” shows an example sequence between a Source and a Sink when the Source gets the Sink’s Manufacturer information for one of its Batteries.

Figure 8-84 “Source Gets Sink’s Battery Manufacturer Information”

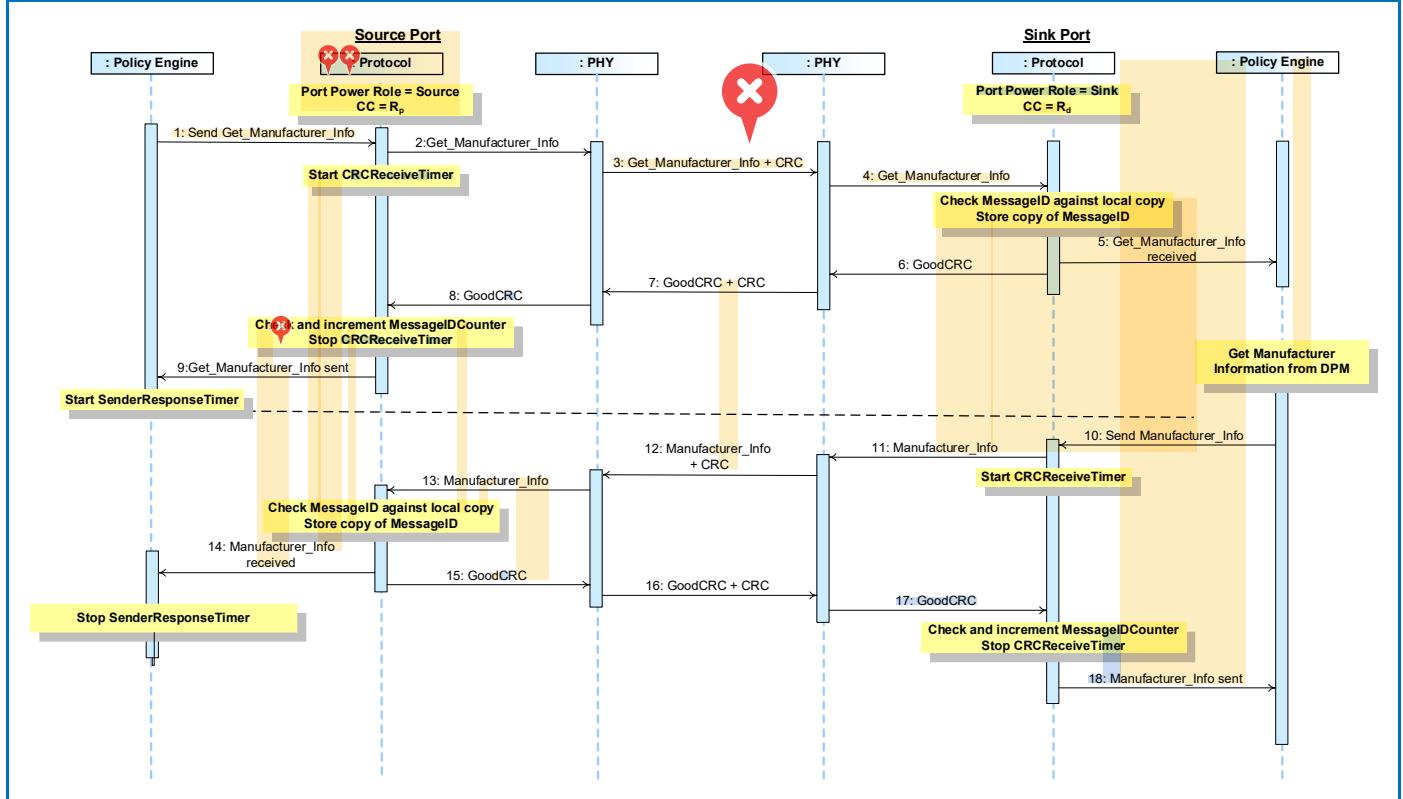


Table 8.112 “Steps for a Source getting Sink’s Battery Manufacturer Information Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-84 “Source Gets Sink’s Battery Manufacturer Information”** above.

Table 8.112 “Steps for a Source getting Sink’s Battery Manufacturer Information Sequence”

Step	Source Port	Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Manufacturer_Info Message with a request for Battery information for a given Battery.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Manufacturer_Info Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Manufacturer_Info Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Manufacturer_Info Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Manufacturer_Info Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Manufacturer_Info Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Battery’s manufacturer information for a given Battery which is provided. The Policy Engine tells the Protocol Layer to form a Manufacturer_Info Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Manufacturer_Info Message.	Physical Layer appends a CRC and sends the Manufacturer_Info Message. Starts CRCReceiveTimer .

13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Manufacturer_Info</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Manufacturer_Info</i> Message was successfully sent.

The Sink has informed the Source of the manufacturer information for the requested Battery.

8.3.2.12.6.4

Sink Gets Battery Manufacturer Information from a Source

Figure 8-85 “Sink Gets Source’s Battery Manufacturer Information” shows an example sequence between a Source and a Sink when the Source gets the Sink’s Manufacturer information for the Port.

Figure 8-85 “Sink Gets Source’s Battery Manufacturer Information”

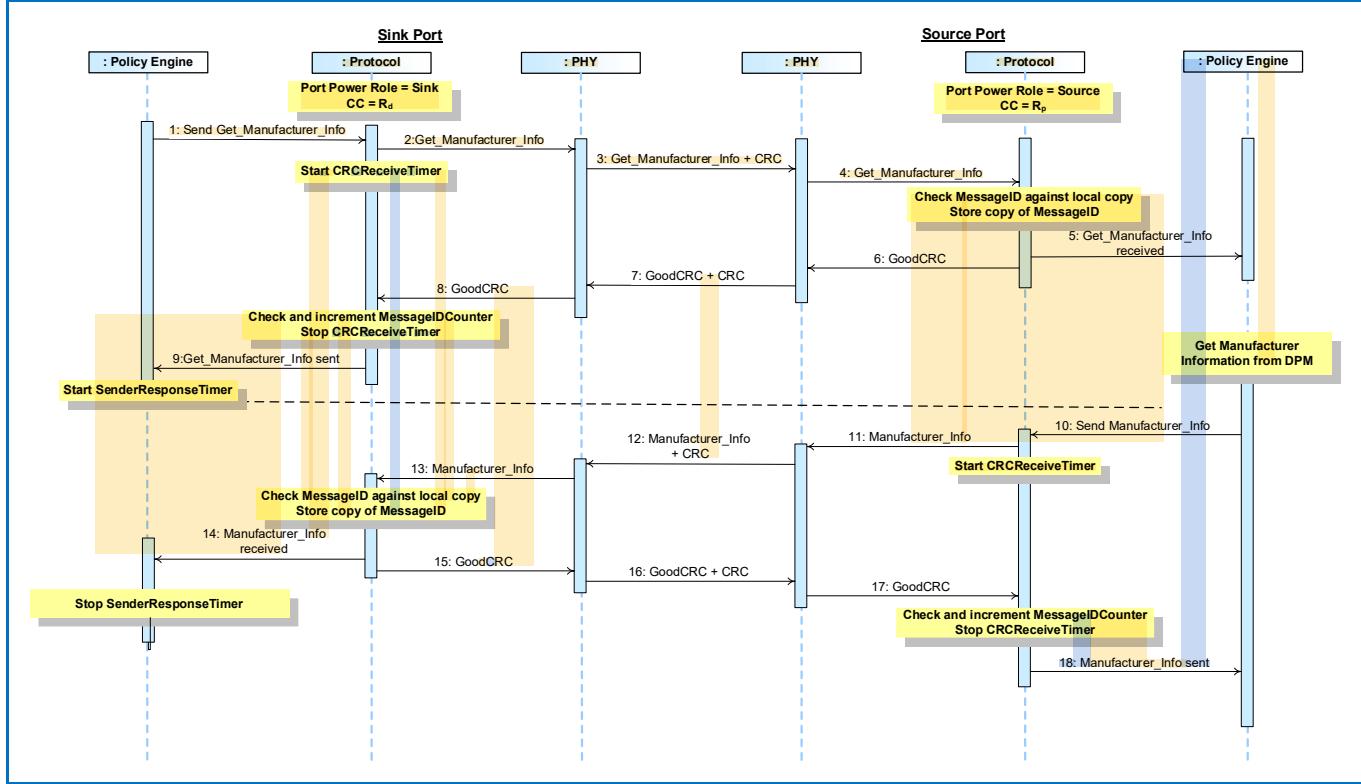


Table 8.113 “Steps for a Source getting Sink’s Battery Manufacturer Information Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-85 “Sink Gets Source’s Battery Manufacturer Information”** above.

Table 8.113 “Steps for a Source getting Sink’s Battery Manufacturer Information Sequence”

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Manufacturer_Info Message with a request for Battery information for a given Battery.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Manufacturer_Info Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Manufacturer_Info Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Manufacturer_Info Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Manufacturer_Info Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Manufacturer_Info Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Battery’s manufacturer information for a given Battery which is provided. The Policy Engine tells the Protocol Layer to form a Manufacturer_Info Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Manufacturer_Info Message.	Physical Layer appends a CRC and sends the Manufacturer_Info Message. Starts CRCReceiveTimer .

13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Manufacturer_Info</i> Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Manufacturer_Info</i> Message was successfully sent.

The Sink has informed the Source of the manufacturer information for the requested Battery.

8.3.2.12.6.5

VCONN Source Gets Manufacturer Information from a Cable Plug

Figure 8-86 “Vconn Source Gets Cable Plug’s Manufacturer Information” shows an example sequence between a VCONN Source (Source or Sink) and a Cable Plug when the VCONN Source gets the Cable Plug’s Manufacturer information.

Figure 8-86 “VCONN Source Gets Cable Plug’s Manufacturer Information”

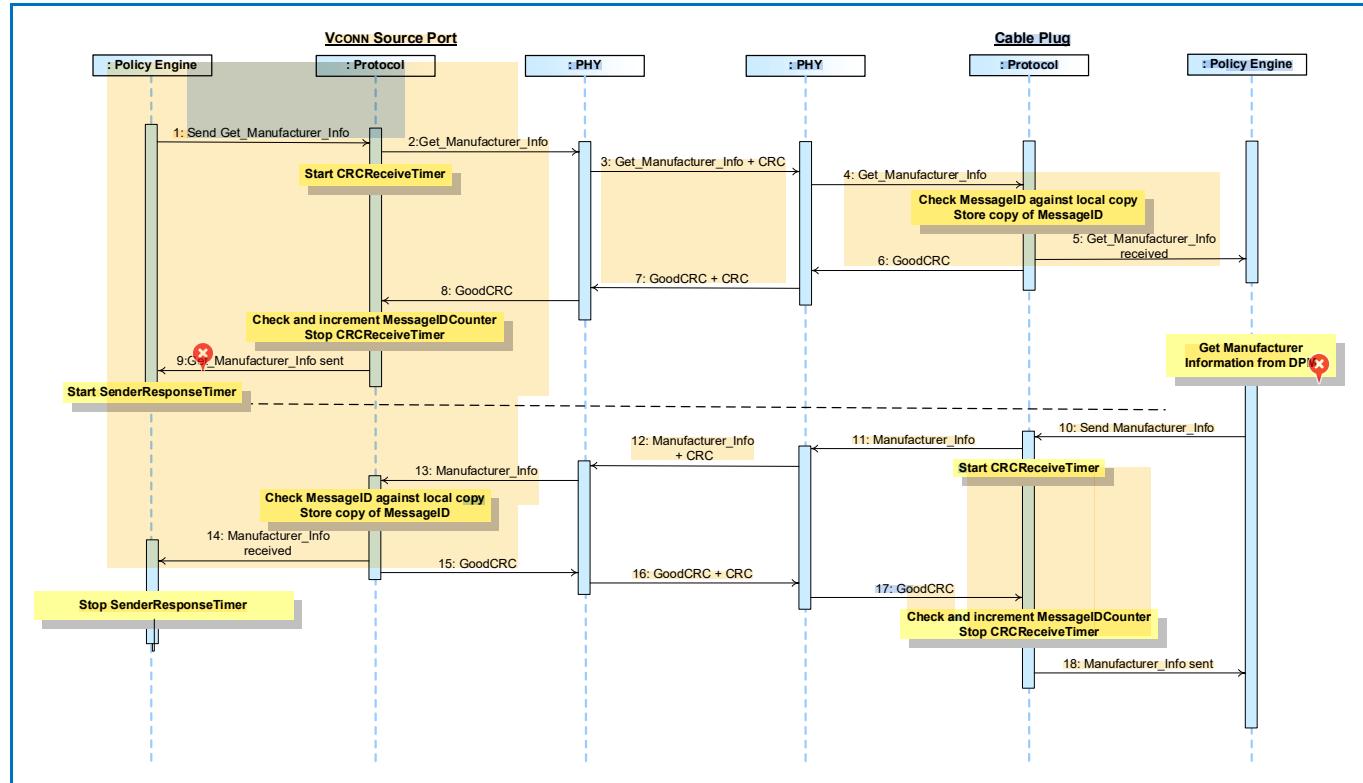


Table 8.114 “Steps for a Vconn Source getting Sink’s Port Manufacturer Information Sequence^x below provides a detailed explanation of what happens at each labeled step in **Figure 8-86 “Vconn Source Gets Cable Plug’s Manufacturer Information”** above.

Table 8.114 “Steps for a VCONN Source getting Sink’s Port Manufacturer Information Sequence”

Step	VCONN Source	Cable Plug
1	The Port is currently acting as the VCONN Source. Policy Engine directs the Protocol Layer to send a Get_Manufacturer_Info Message with a request for Port information.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Manufacturer_Info Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Manufacturer_Info Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Manufacturer_Info Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Manufacturer_Info Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Manufacturer_Info Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Cable Plug’s manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a Manufacturer_Info Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Manufacturer_Info Message.	Physical Layer appends a CRC and sends the Manufacturer_Info Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Manufacturer_Info Message information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Manufacturer_Info</i> Message was successfully sent.
The Cable Plug has informed the Source of its manufacturer information.		

8.3.2.12.7

Country Codes

8.3.2.12.7.1

Source Gets Country Codes from a Sink

Figure 8-87 “Source Gets Sink’s Country Codes” shows an example sequence between a Source and a Sink when the Source gets the Sink’s Country Codes.

Figure 8-87 “Source Gets Sink’s Country Codes”

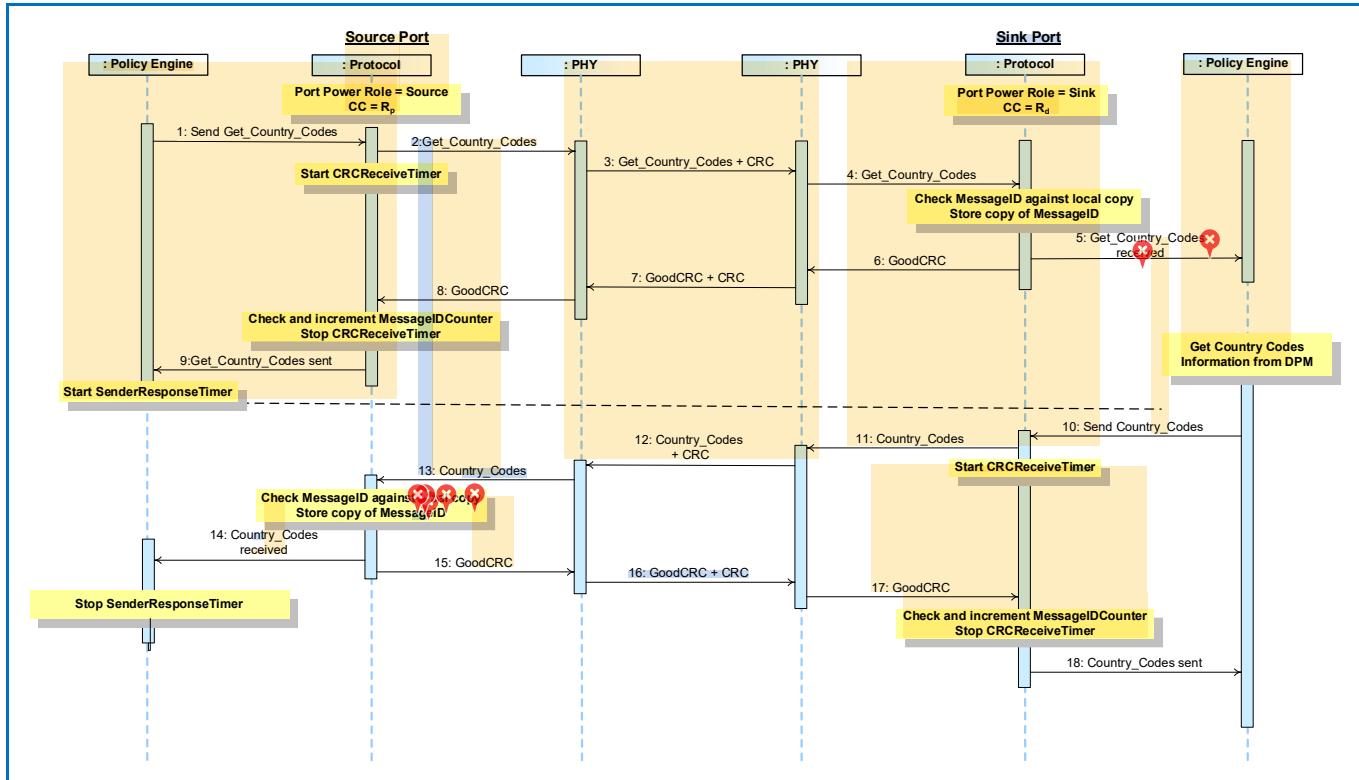


Table 8.115 “Steps for a Source getting Country Codes Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-87 “Source Gets Sink’s Country Codes”** above.

Table 8.115 “Steps for a Source getting Country Codes Sequence”

Step	Source Port	Sink Port
1	The Port has Port Power Role set to Source and the Rp pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Country_Codes Message with a request for Port information. 	The Port has Port Power Role set to Sink with the Rd pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the  Get_Country_Codes Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Country_Codes Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Country_Codes Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Country_Codes Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the  GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Country_Codes Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Port’s manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a Country_Codes Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Country_Codes Message.	Physical Layer appends a CRC and sends the Country_Codes Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Country_Codes Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC?</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Country_Codes</i> Message was successfully sent.
The Sink has informed the Source of the country codes.		

8.3.2.12.7.2 Sink Gets Country Codes from a Source

Figure 8-88 “Sink Gets Source’s Country Codes” shows an example sequence between a Source and a Sink when the Source gets the Sink’s country codes.

Figure 8-88 “Sink Gets Source’s Country Codes”

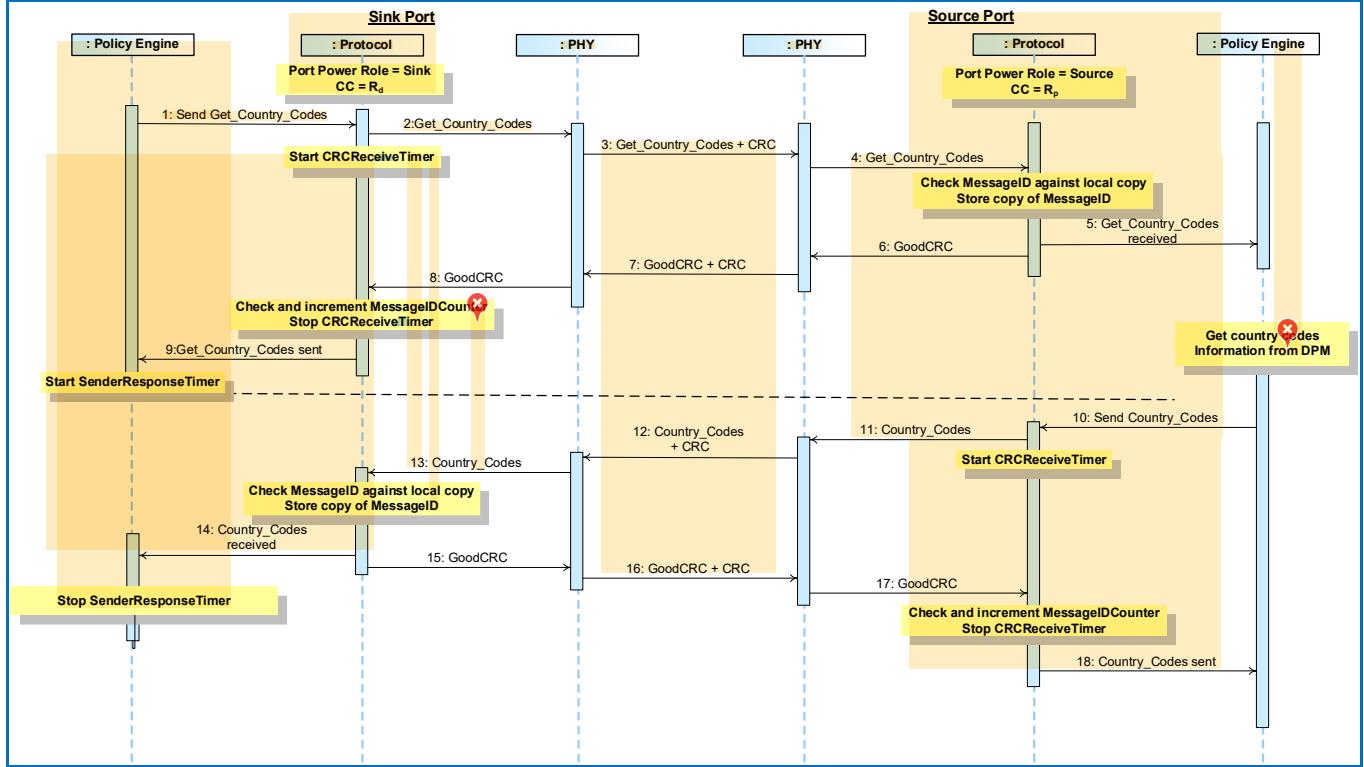


Table 8.116 “Steps for a Source getting Sink’s Country Codes Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-88 “Sink Gets Source’s Country Codes”** above.

Table 8.116 “Steps for a Source getting Sink’s Country Codes Sequence”

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Country_Codes Message with a request for Port information. \times	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Country_Codes Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Country_Codes Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Country_Codes Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Country_Codes Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Country_Codes Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Port’s manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a Country_Codes Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Country_Codes Message.	Physical Layer appends a CRC and sends the Country_Codes Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Country_Codes Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Country_Codes</i> Message was successfully sent.
The Sink has informed the Source of the country codes.		

8.3.2.12.7.3

VCONN Source Gets Country Codes from a Cable Plug

Figure 8-89 “Vconn Source Gets Cable Plug’s Country Codes” shows an example sequence between a VCONN Source (Source or Sink) and a Cable Plug when the VCONN Source gets the Cable Plug’s Country Codes.

Figure 8-89 “VCONN Source Gets Cable Plug’s Country Codes”

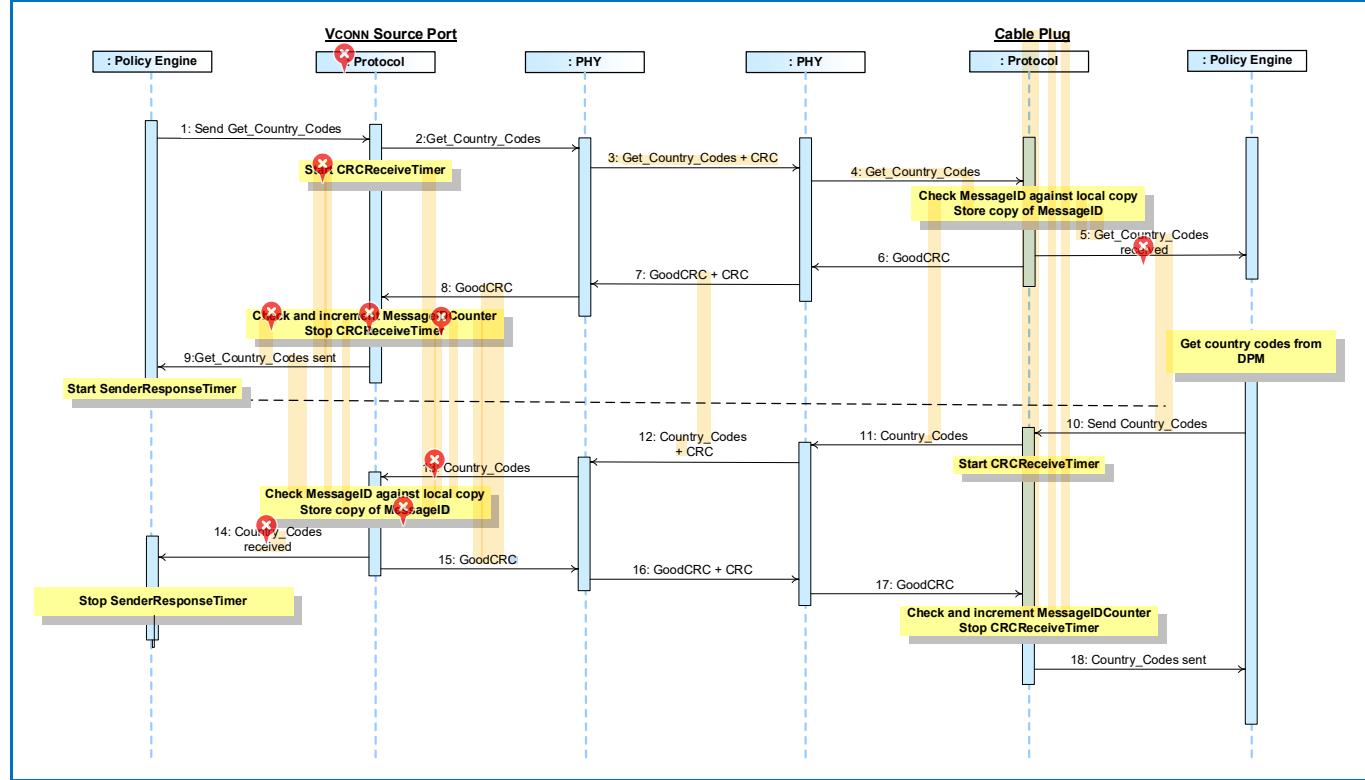


Table 8.117 "Steps for a Vconn Source getting Sink's Country Codes Sequence" below provides a detailed explanation of what happens at each labeled step in **Figure 8-89 "Vconn Source Gets Cable Plug's Country Codes"** above.

Table 8.117 "Steps for a VCONN Source getting Sink's Country Codes Sequence"

Step	VCONN Source	Cable Plug
1	The Port is currently acting as the VCONN Source. Policy Engine directs the Protocol Layer to send a Get_Country_Codes Message with a request for Port information.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Country_Codes Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Country_Codes Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Country_Codes Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Country_Codes Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Country_Codes Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Cable Plug's manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a Country_Codes Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Country_Codes Message.	Physical Layer appends a CRC and sends the Country_Codes Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Country_Codes Message information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>SenderResponseTimer</i> .	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Country_Codes</i> Message was successfully sent.
The Cable Plug has informed the Source of its country codes.		

8.3.2.12.8

Country Information

8.3.2.12.8.1

Source Gets Country Information from a Sink

Figure 8-90 “Source Gets Sink’s Country Information” shows an example sequence between a Source and a Sink when the Source gets the Sink’s country information.

Figure 8-90 “Source Gets Sink’s Country Information”

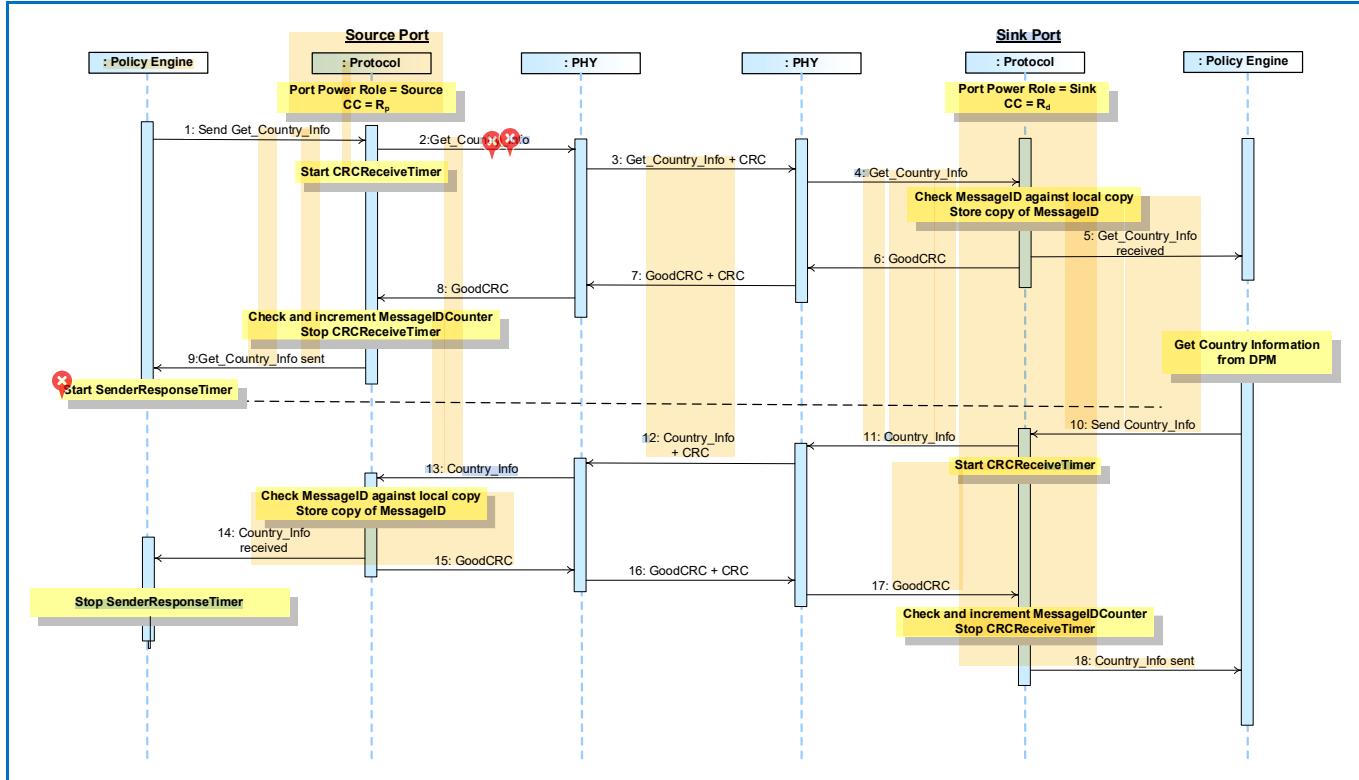


Table 8.118 “Steps for a Source getting Country Information Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-90 “Source Gets Sink’s Country Information”** above.

Table 8.118 “Steps for a Source getting Country Information Sequence”

Step	Source Port	Sink Port
1	The Port has Port Power Role set to Source and the Rp pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Country_Info Message with a request for Port information for a specific Country Code.📍	The Port has Port Power Role set to Sink with the Rd pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Country_Info Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Country_Info Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Country_Info Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Country_Info Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.📍	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Country_Info Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Port’s manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a Country_Info Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Country_Info Message.	Physical Layer appends a CRC and sends the Country_Info Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Country_Info Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Country_Info</i> Message was successfully sent.
The Sink has informed the Source of the country information.		

8.3.2.12.8.2

Sink Gets Country Information from a Source

Figure 8-91 “Sink Gets Source’s Country Information” shows an example sequence between a Source and a Sink when the Source gets the Sink’s country codes.

Figure 8-91 “Sink Gets Source’s Country Information”

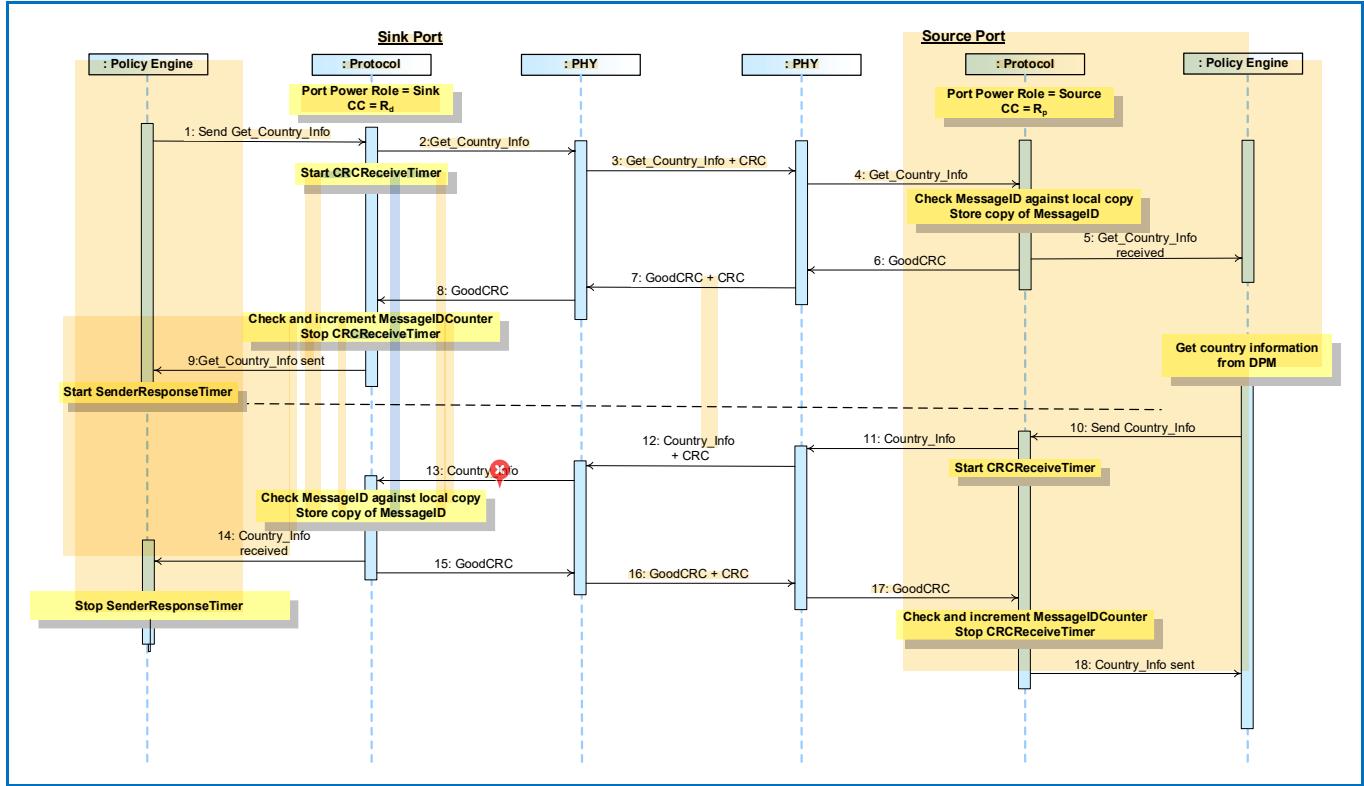


Table 8.119 “Steps for a Source getting Sink’s Country Information Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-91 “Sink Gets Source’s Country Information”** above.

Table 8.119 “Steps for a Source getting Sink’s Country Information Sequence”

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Country_Info Message with a request for Port information for a specific country code.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Country_Info Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Country_Info Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Country_Info Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Country_Info Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Country_Info Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Port’s manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a Country_Info Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Country_Info Message.	Physical Layer appends a CRC and sends the Country_Info Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Country_Info Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC?</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Country_Info</i> Message was successfully sent.
The Sink has informed the Source of the country information.		

8.3.2.12.8.3

VCONN Source Gets Country Information from a Cable Plug

Figure 8-92 “Vconn Source Gets Cable Plug’s Country Information” shows an example sequence between a VCONN Source (Source or Sink) and a Cable Plug when the VCONN Source gets the Cable Plug’s country information.

Figure 8-92 “VCONN Source Gets Cable Plug’s Country Information”

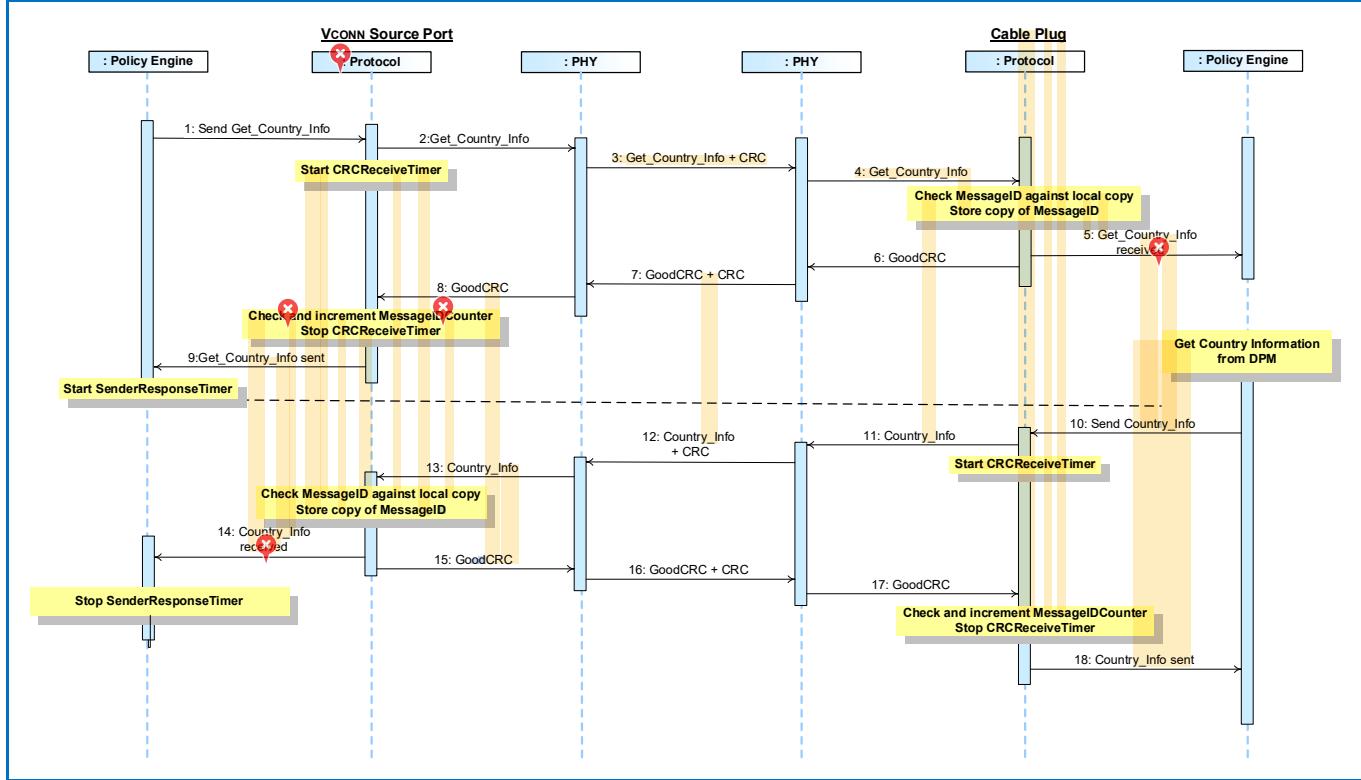


Table 8.120 “Steps for a Vconn Source getting Sink’s Country Information Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-92 “Vconn Source Gets Cable Plug’s Country Information”** above.

Table 8.120 “Steps for a VCONN Source getting Sink’s Country Information Sequence”

Step	VCONN Source	Cable Plug
1	The Port is currently acting as the VCONN Source. Policy Engine directs the Protocol Layer to send a Get_Country_Info Message with a request for Port information for a specific country code.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Country_Info Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Country_Info Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Country_Info Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Country_Info Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Country_Info Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Cable Plug’s manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a Country_Info Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Country_Info Message.	Physical Layer appends a CRC and sends the Country_Info Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Country_Info Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Country_Info</i> Message was successfully sent.
The Cable Plug has informed the Source of its country information.		

8.3.2.12.9 Revision Information

8.3.2.12.9.1 Source Gets Revision Information from a Sink

Figure 8-93 “Source Gets Sink’s Revision Information” shows an example sequence between a Source and a Sink when the Source gets the Sink’s Revision information.

Figure 8-93 “Source Gets Sink’s Revision Information”

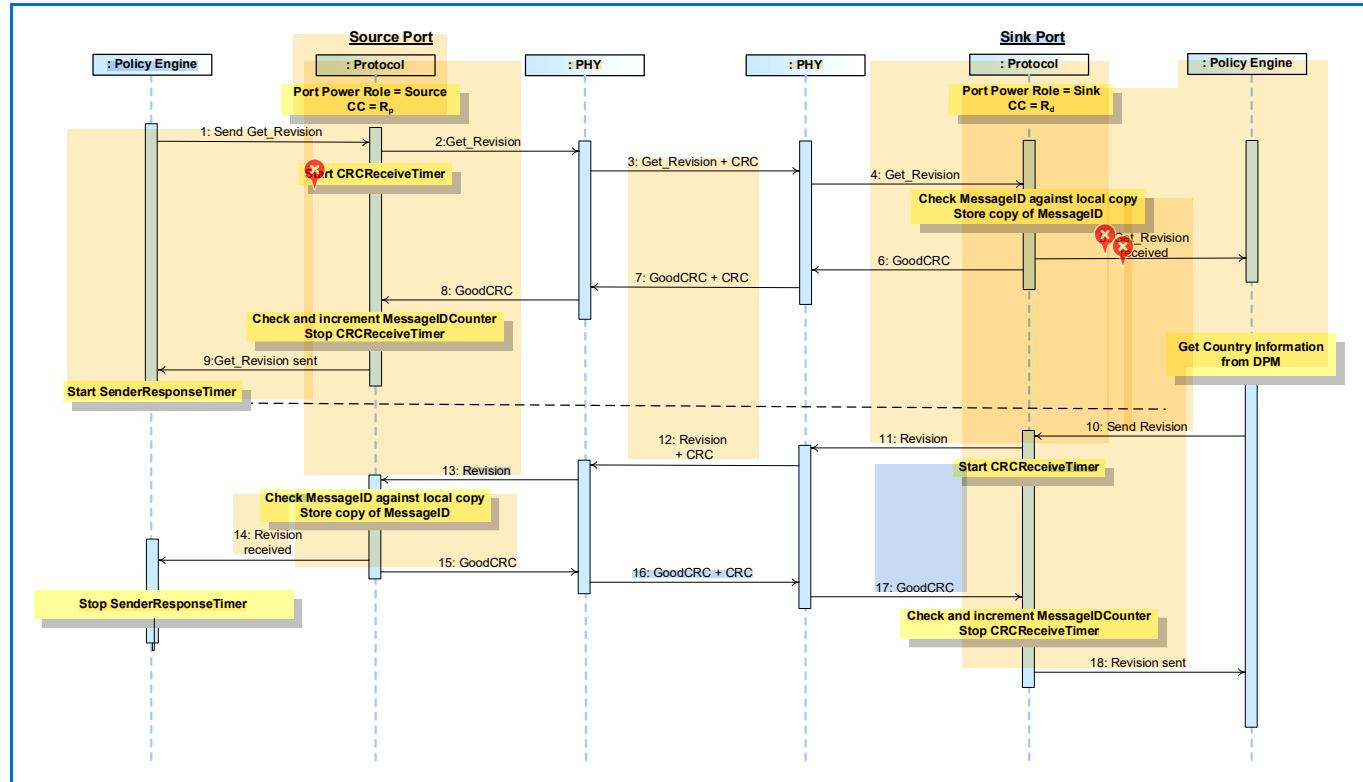


Table 8.121 “Steps for a Source getting Revision Information Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-93 “Source Gets Sink’s Revision Information”** above.

Table 8.121 “Steps for a Source getting Revision Information Sequence”

Step	Source Port	Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Revision Message with a request for Port information for a specific Revision Code. 	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the  Get_Revision Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Revision Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Revision Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Revision Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Revision Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Port’s manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a Revision Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Revision_Info Message.	Physical Layer appends a CRC and sends the Revision Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Revision Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Revision</i> Message was successfully sent.
The Sink has informed the Source of the Revision information.		

8.3.2.12.9.2

Sink Gets Revision Information from a Source

Figure 8-94 “Sink Gets Source’s Revision Information” shows an example sequence between a Source and a Sink when the Source gets the Sink’s Revision codes.

Figure 8-94 “Sink Gets Source’s Revision Information”

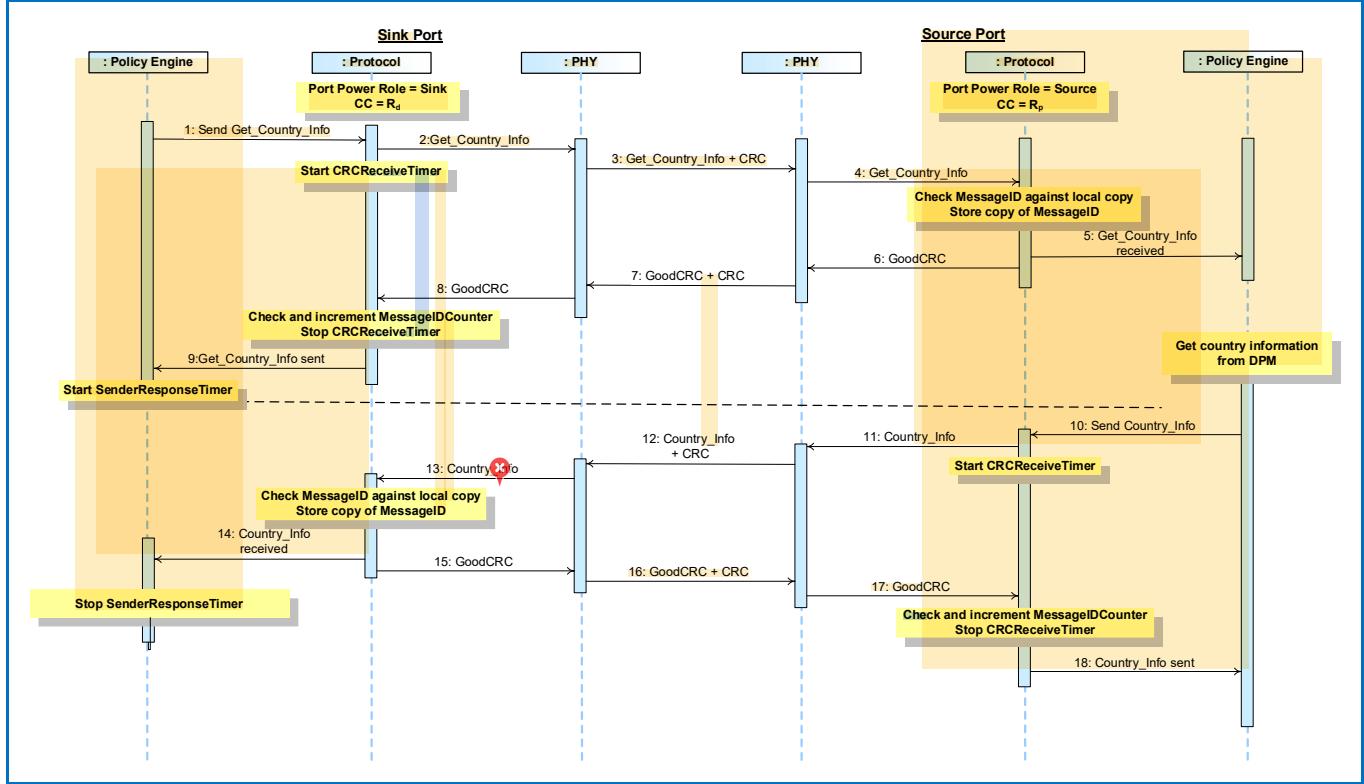


Table 8.122 “Steps for a Source getting Sink’s Revision Information Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-94 “Sink Gets Source’s Revision Information”** above.

Table 8.122 “Steps for a Source getting Sink’s Revision Information Sequence”

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Revision Message with a request for Port information for a specific Revision code.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Revision Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Revision Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Revision Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Revision Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Revision Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Port’s manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a Revision Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Revision_Info Message.	Physical Layer appends a CRC and sends the Revision Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Revision Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC?</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Revision</i> Message was successfully sent.
The Sink has informed the Source of the Revision information.		

8.3.2.12.9.3

VCONN Source Gets Revision Information from a Cable Plug

Figure 8-95 “Vconn Source Gets Cable Plug’s Revision Information” shows an example sequence between a VCONN Source (Source or Sink) and a Cable Plug when the VCONN Source gets the Cable Plug’s Revision information.

Figure 8-95 “VCONN Source Gets Cable Plug’s Revision Information”

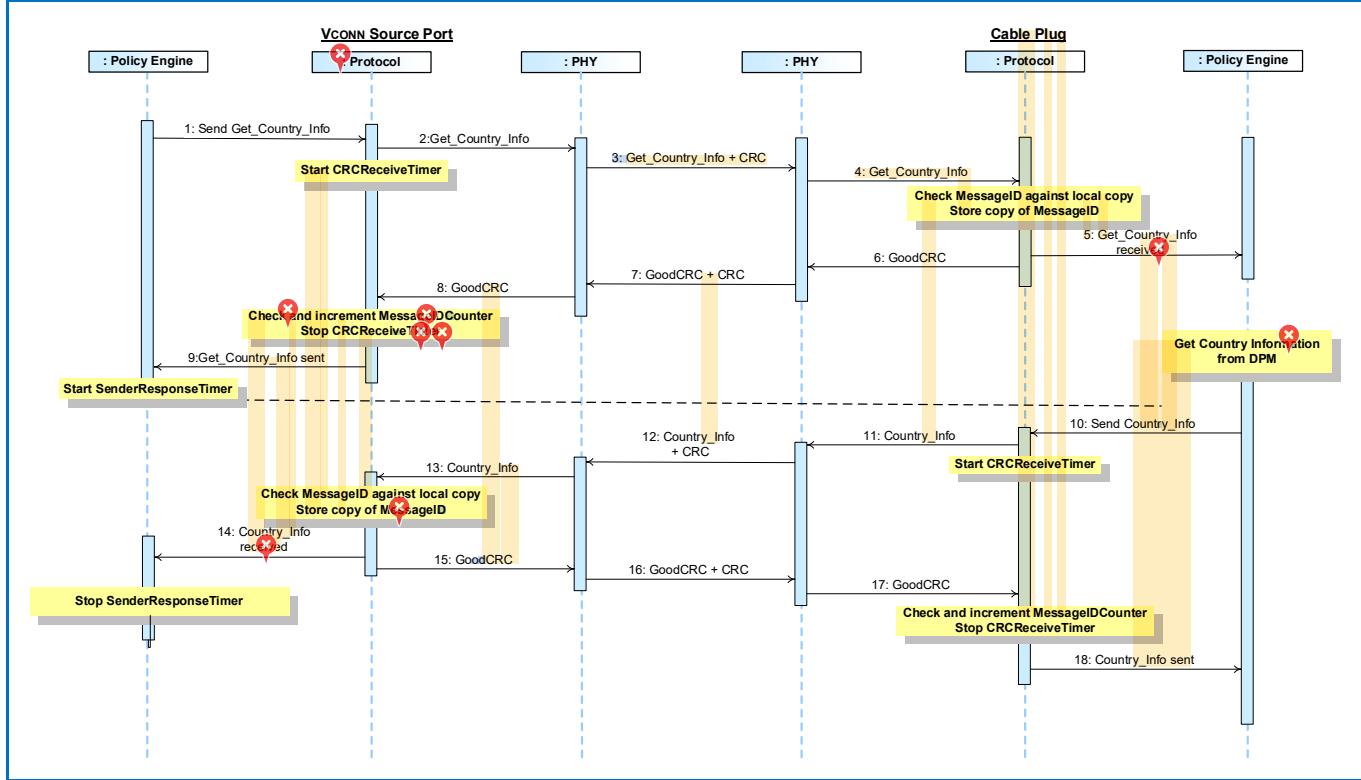


Table 8.123 "Steps for a Vconn Source getting Sink's Revision Information Sequence" below provides a detailed explanation of what happens at each labeled step in **Figure 8-95 "Vconn Source Gets Cable Plug's Revision Information"** above.

Table 8.123 "Steps for a VCONN Source getting Sink's Revision Information Sequence"

Step	VCONN Source	Cable Plug
1	The Port is currently acting as the VCONN Source. Policy Engine directs the Protocol Layer to send a Get_Revision Message with a request for Port information for a specific Revision code.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Revision Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Revision Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Revision Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Revision Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Revision Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the Cable Plug's manufacturer information which is provided. The Policy Engine tells the Protocol Layer to form a Revision Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Revision Message.	Physical Layer appends a CRC and sends the Revision Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Revision Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Revision</i> Message was successfully sent.
The Cable Plug has informed the Source of its Revision information.		

8.3.2.12.10 Source Information

8.3.2.12.10.1 Sink Gets Source Information

Figure 8-96 “Sink Gets Source’s Information” shows an example sequence between a Source and a Sink when the Sink gets the Source’s information.

Figure 8-96 “Sink Gets Source’s Information”

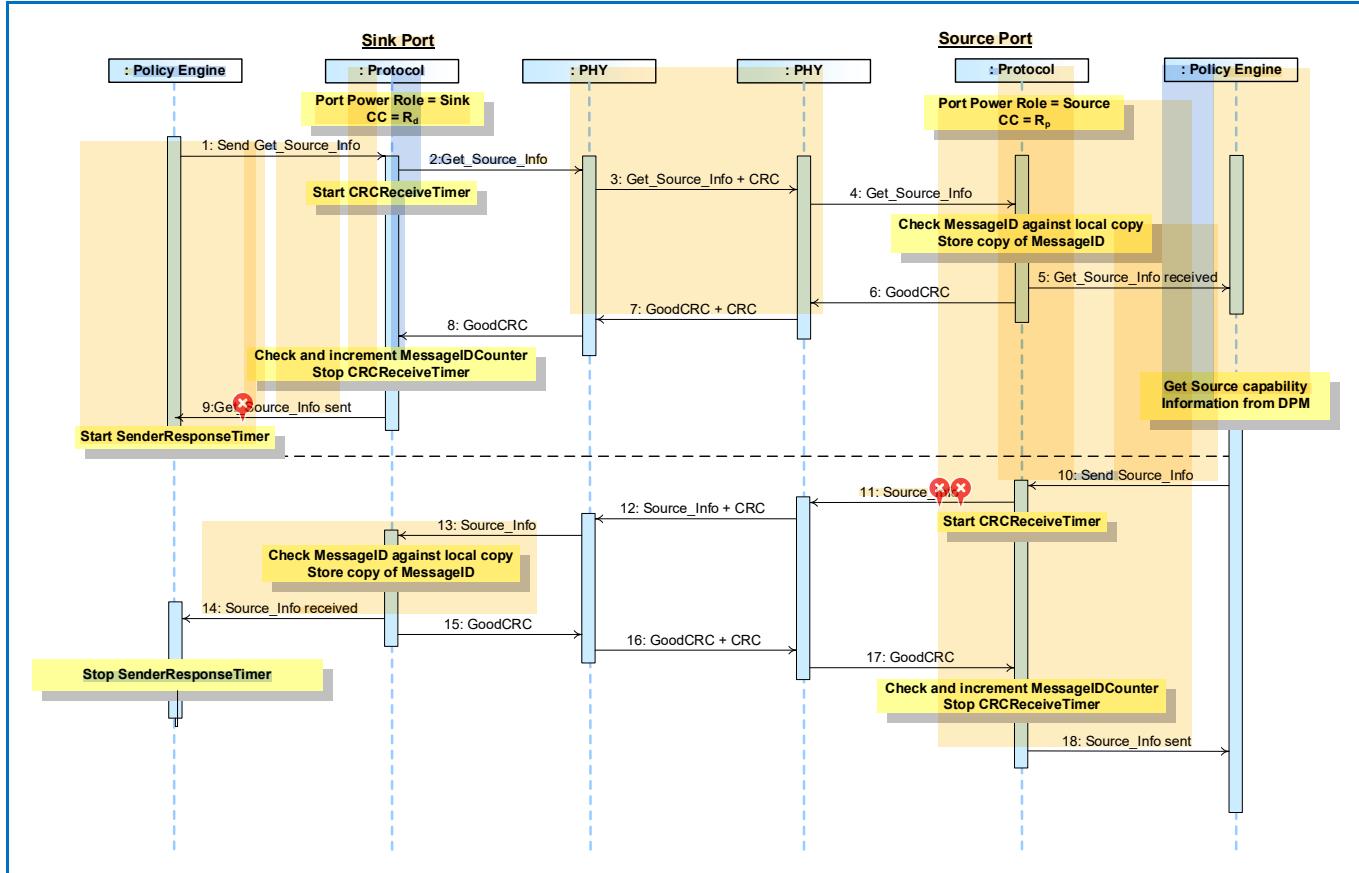


Table 8.124 "Steps for a Sink getting Source Information Sequence" below provides a detailed explanation of what happens at each labeled step in **Figure 8-96 "Sink Gets Source's Information"** above.

Table 8.124 "Steps for a Sink getting Source Information Sequence"

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Source_Info Message.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Source_Info Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Source_Info Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Source_Info Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Source_Info Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Source_Info Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Source information which is provided. The Policy Engine tells the Protocol Layer to form a Source_Info Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Source_Info Message.	Physical Layer appends a CRC and sends the Source_Info Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Source_Info Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Info</i> Message was successfully sent.
The Source has provided the Sink with its information.		

8.3.2.12.10.2

Dual-Role Source Gets Source Information from a Dual-Role Sink

Figure 8-97 “Dual-Role Source Gets Dual-Role Sink’s Information as a Source” shows an example sequence between a Dual-Role Source and a Dual-Role Sink when the Source gets the Sink’s Information as a Source.

Figure 8-97 “Dual-Role Source Gets Dual-Role Sink’s Information as a Source”

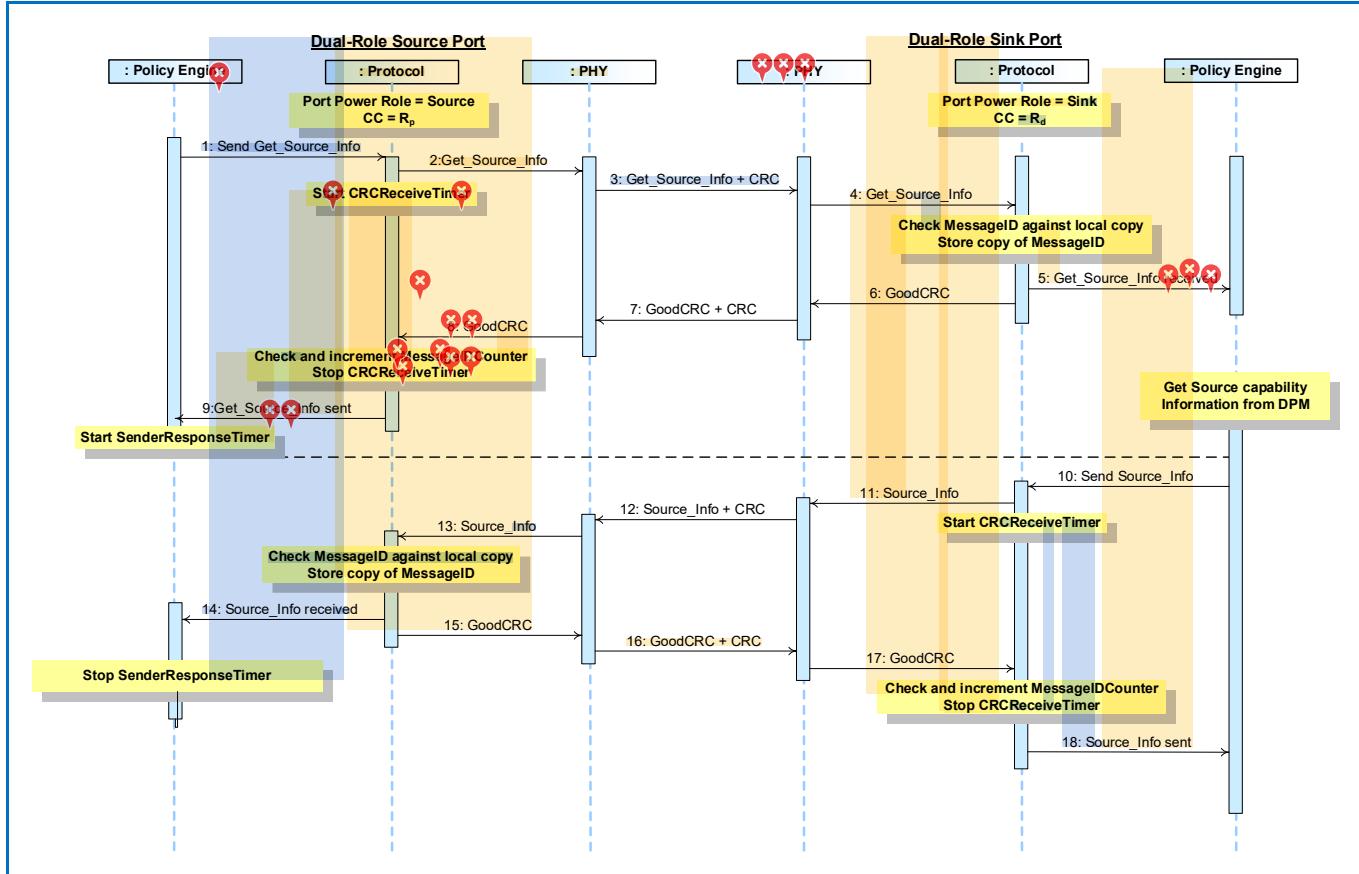


Table 8.125 “Steps for a Dual-Role Source getting Dual-Role Sink’s Information as a Source Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-97 “Dual-Role Source Gets Dual-Role Sink’s Information as a Source”** above.

Table 8.125 “Steps for a Dual-Role Source getting Dual-Role Sink’s Information as a Source Sequence”

Step	Dual-Role Source Port	Dual-Role Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Get_Source_Info Message.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Get_Source_Info Message. Starts CRCReceiveTimer .	Physical Layer receives the Get_Source_Info Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Get_Source_Info Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Get_Source_Info Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Get_Source_Info Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine requests the DPM for the present Source information which is provided. The Policy Engine tells the Protocol Layer to form a Source_Info Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Source_Info Message.	Physical Layer appends a CRC and sends the Source_Info Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Source_Info Message information to the Policy Engine that consumes it.	
14	The Policy Engine stops the SenderResponseTimer .	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Source_Info</i> Message was successfully sent.
The Dual-Role Sink has provided the Dual-Role Source with its information.		

8.3.2.13 Security

8.3.2.13.1 Source requests security exchange with Sink

Figure 8-98 “Source requests security exchange with Sink” shows an example sequence for a security exchange between a Source and a Sink.

Figure 8-98 “Source requests security exchange with Sink”

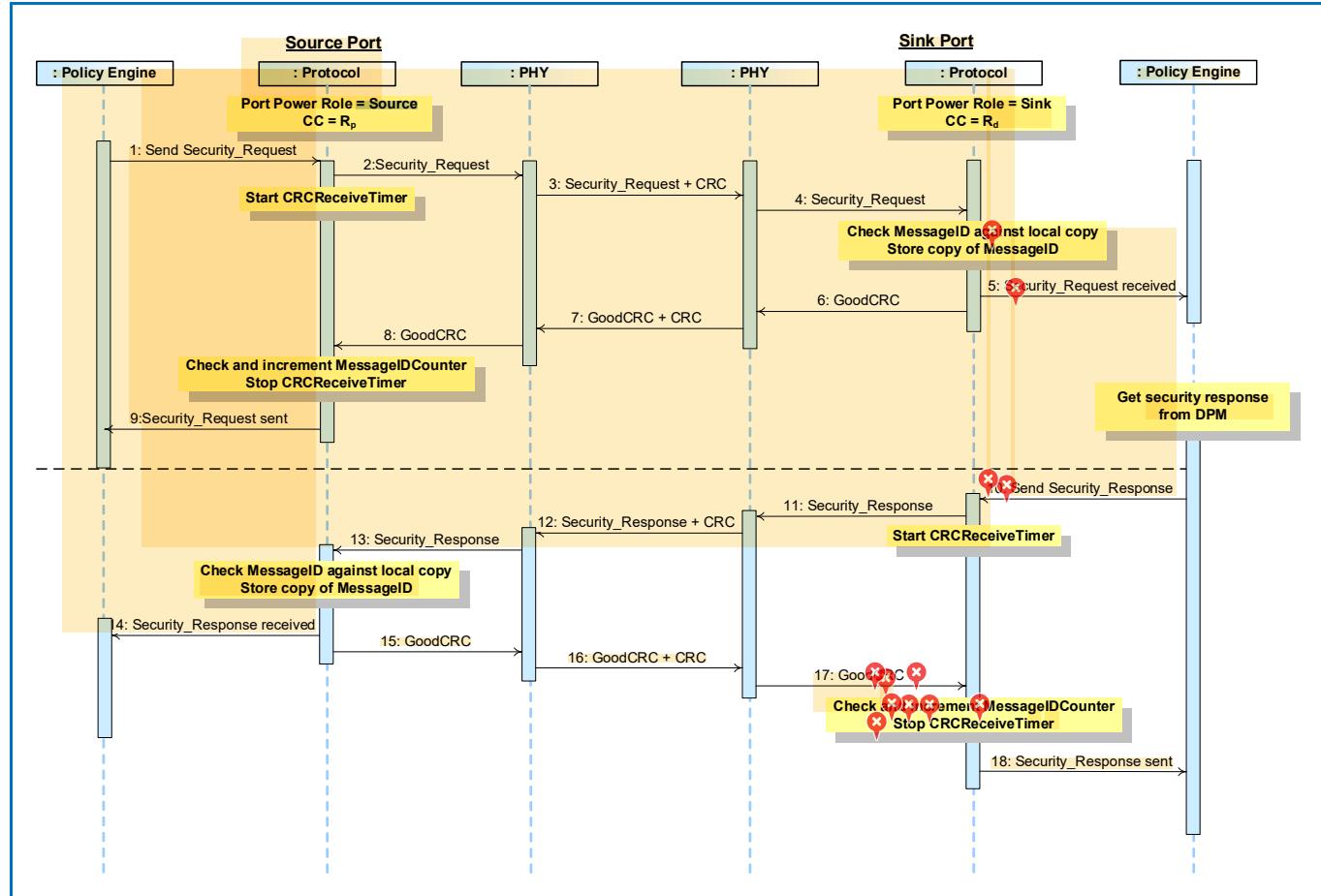


Table 8.126 “Steps for a Source requesting a security exchange with a Sink Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-98 “Source requests security exchange with Sink”** above.

Table 8.126 “Steps for a Source requesting a security exchange with a Sink Sequence”

Step	Source Port	Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Security_Request Message using a payload supplied by the DPM.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Security_Request Message. Starts CRCReceiveTimer .	Physical Layer receives the Security_Request Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Security_Request Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Security_Request Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Security_Request Message was successfully sent.	
10		Policy Engine requests the DPM for the response to the security request which is provided. The Policy Engine tells the Protocol Layer to form a Security_Response Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Security_Response Message.	Physical Layer appends a CRC and sends the Security_Response Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Security_Response Message information to the Policy Engine that consumes it.	

14	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
15	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
16		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
17		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Security_Response</i> Message was successfully sent.
The security exchange is complete.		

8.3.2.13.2 Sink requests security exchange with Source

Figure 8-99 “Sink requests security exchange with Source” shows an example sequence for a security exchange between a Sink and a Source.

Figure 8-99 “Sink requests security exchange with Source”

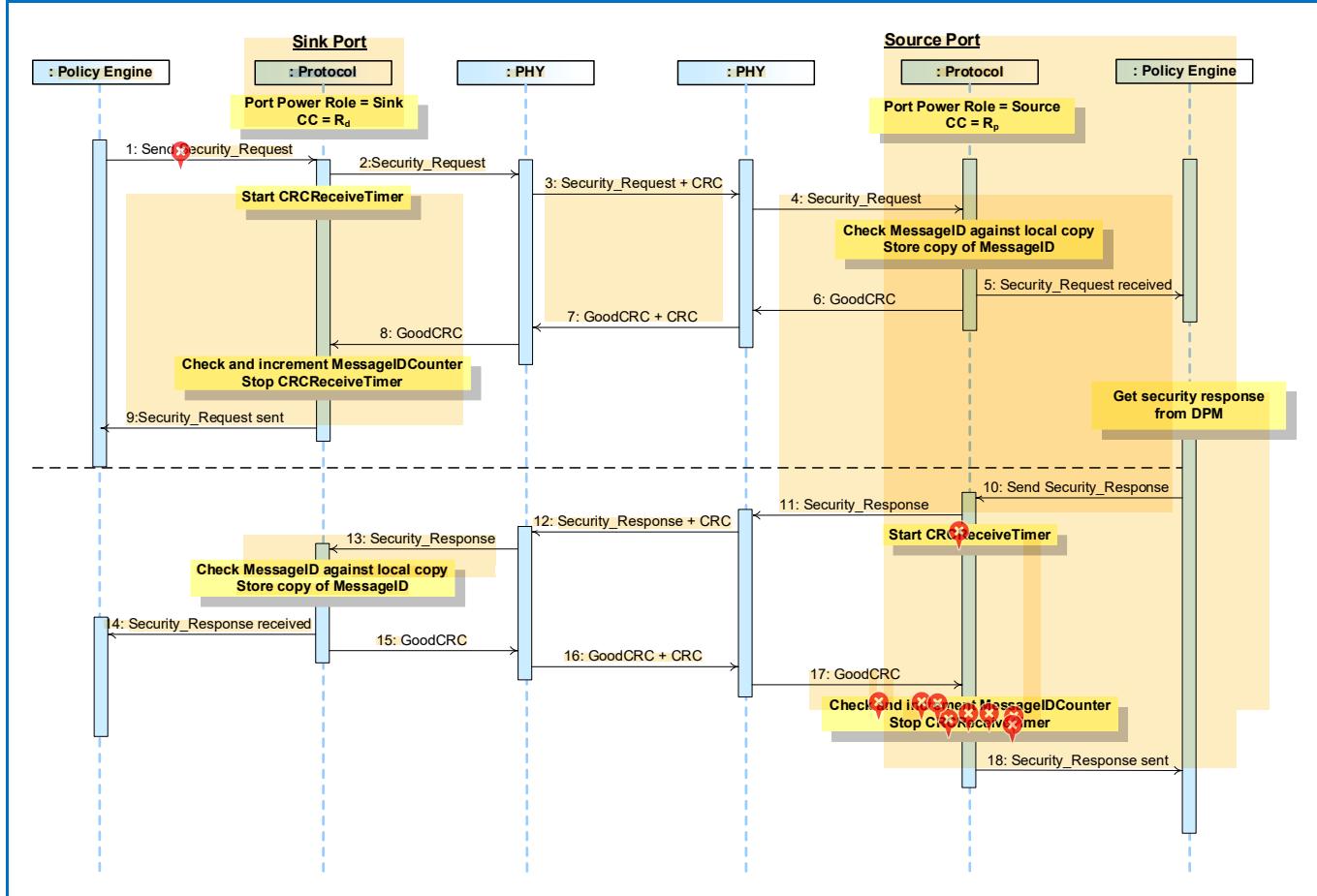


Table 8.127 "Steps for a Sink requesting a security exchange with a Source Sequence" below provides a detailed explanation of what happens at each labeled step in **Figure 8-99 "Sink requests security exchange with Source"** above.

Table 8.127 "Steps for a Sink requesting a security exchange with a Source Sequence"

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Security_Request Message using a payload supplied by the DPM.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Security_Request Message. Starts CRCReceiveTimer .	Physical Layer receives the Security_Request Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Security_Request Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Security_Request Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Security_Request Message was successfully sent.	
10		Policy Engine requests the DPM for the response to the security request which is provided. The Policy Engine tells the Protocol Layer to form a Security_Response Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Security_Response Message.	Physical Layer appends a CRC and sends the Security_Response Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Security_Response Message information to the Policy Engine that consumes it.	

14	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
15	Physical Layer appends a CRC and sends the <i>GoodCRC?</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
16		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
17		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Security_Response</i> Message was successfully sent.
The security exchange is complete.		

8.3.2.13.3

VCONN Source requests security exchange with Cable Plug

Figure 8-100 “Vconn Source requests security exchange with Cable Plug” shows an example sequence for a security exchange between a VCONN Source and a Cable Plug.

Figure 8-100 “Vconn Source requests security exchange with Cable Plug”

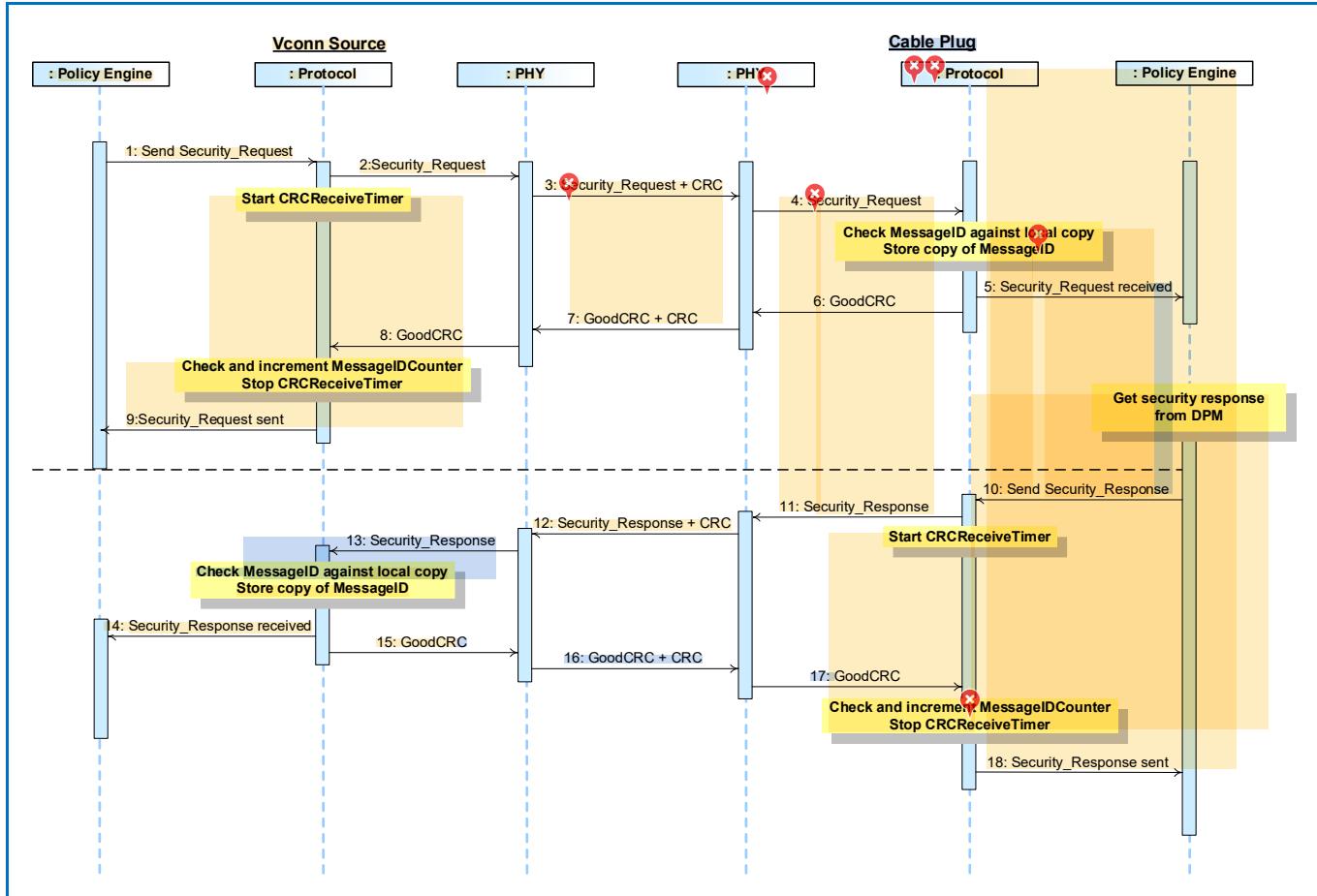


Table 8.128 "Steps for a Vconn Source requesting a security exchange with a Cable Plug Sequence" below provides a detailed explanation of what happens at each labeled step in **Figure 8-100 "Vconn Source requests security exchange with Cable Plug"** above.

Table 8.128 "Steps for a VCONN Source requesting a security exchange with a Cable Plug Sequence"

Step	VCONN Source	Cable Plug
1	Policy Engine directs the Protocol Layer to send a Security_Request Message using a payload supplied by the DPM.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Security_Request Message. Starts CRCReceiveTimer .	Physical Layer receives the Security_Request Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Security_Request Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Security_Request Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Security_Request Message was successfully sent.	
10		Policy Engine requests the DPM for the response to the security request which is provided. The Policy Engine tells the Protocol Layer to form a Security_Response Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Security_Response Message.	Physical Layer appends a CRC and sends the Security_Response Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Security_Response Message information to the Policy Engine that consumes it.	
14	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	

15	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
16		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
17		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Security_Response</i> Message was successfully sent.
The security exchange is complete.		

8.3.2.14 Firmware Update

8.3.2.14.1 Source requests firmware update exchange with Sink

Figure 8-101 “Source requests firmware update exchange with Sink” shows an example sequence for a firmware update exchange between a Source and a Sink.

Figure 8-101 “Source requests firmware update exchange with Sink”

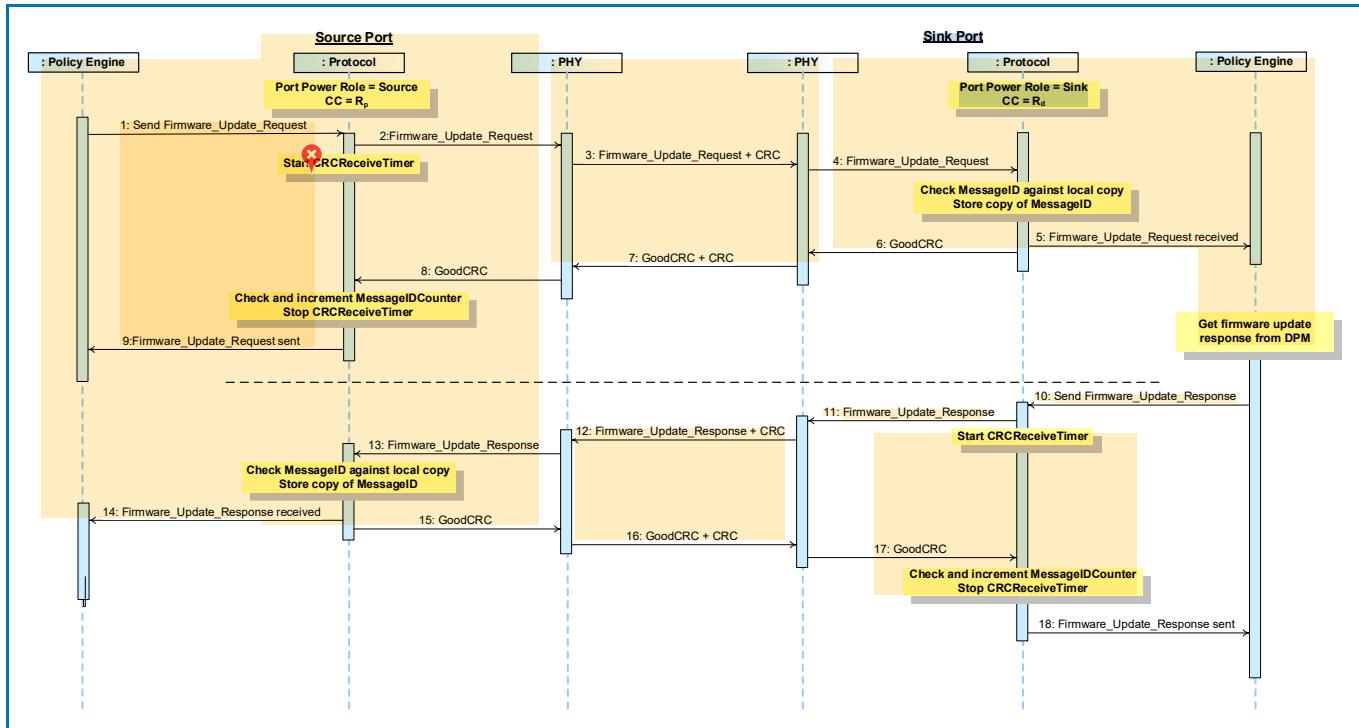


Table 8.129 “Steps for a Source requesting a firmware update exchange with a Sink Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-101 “Source requests firmware update exchange with Sink”** above.

Table 8.129 “Steps for a Source requesting a firmware update exchange with a Sink Sequence”

Step	Source Port	Sink Port
1	The Port has Port Power Role set to Source and the R_p pull up on its CC wire. Policy Engine directs the Protocol Layer to send a Firmware_Update_Request Message using a payload supplied by the DPM.	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Firmware_Update_Request Message. Starts CRCReceiveTimer .	Physical Layer receives the Firmware_Update_Request Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Firmware_Update_Request Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Firmware_Update_Request Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Firmware_Update_Request Message was successfully sent.	
10		Policy Engine requests the DPM for the response to the firmware update request which is provided. The Policy Engine tells the Protocol Layer to form a Firmware_Update_Response Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Firmware_Update_Response Message.	Physical Layer appends a CRC and sends the Firmware_Update_Response Message. Starts CRCReceiveTimer .

13	<p>Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value.</p> <p>The Protocol Layer forwards the received <i>Firmware_Update_Response</i> Message information to the Policy Engine that consumes it.</p>	
14	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
15	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
16		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
17		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Firmware_Update_Response</i> Message was successfully sent.
The firmware update exchange is complete.		

8.3.2.14.2

Sink requests firmware update exchange with Source

Figure 8-102 “Sink requests firmware update exchange with Source” shows an example sequence for a firmware update exchange between a Sink and a Source.

Figure 8-102 “Sink requests firmware update exchange with Source”

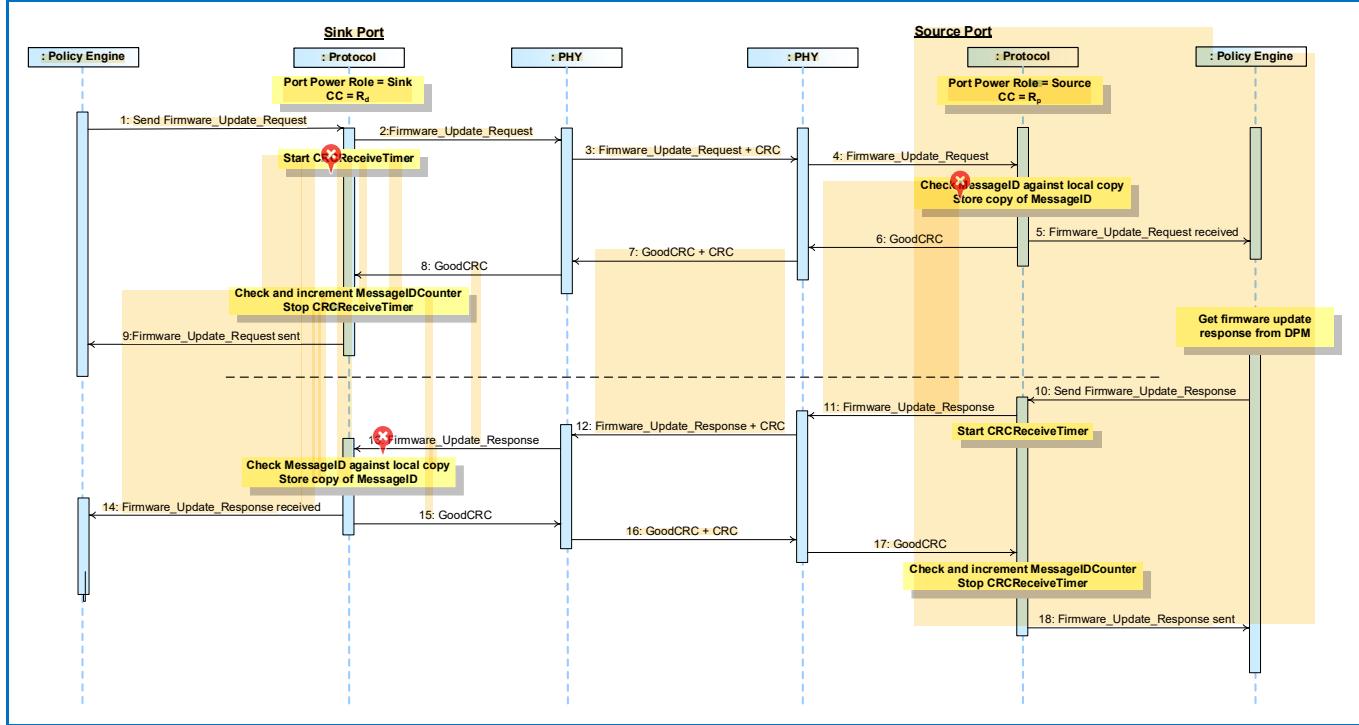


Table 8.130 “Steps for a Sink requesting a firmware update exchange with a Source Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-102 “Sink requests firmware update exchange with Source”** above.

Table 8.130 “Steps for a Sink requesting a firmware update exchange with a Source Sequence”

Step	Sink Port	Source Port
1	The Port has Port Power Role set to Sink with the R_d pull down on its CC wire. Policy Engine directs the Protocol Layer to send a Firmware_Update_Request Message using a payload supplied by the DPM.	The Port has Port Power Role set to Source and the R_p pull up on its CC wire.
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Firmware_Update_Request Message. Starts CRCReceiveTimer .	Physical Layer receives the Firmware_Update_Request Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Firmware_Update_Request Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Firmware_Update_Request Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Firmware_Update_Request Message was successfully sent.	
10		Policy Engine requests the DPM for the response to the firmware update request which is provided. The Policy Engine tells the Protocol Layer to form a Firmware_Update_Response Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Firmware_Update_Response Message.	Physical Layer appends a CRC and sends the Firmware_Update_Response Message. Starts CRCReceiveTimer .

13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Firmware_Update_Response</i> Message information to the Policy Engine that consumes it.	
14	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
15	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
16		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
17		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Firmware_Update_Response</i> Message was successfully sent.
The firmware update exchange is complete.		

8.3.2.14.3

VCONN Source requests firmware update exchange with Cable Plug

Figure 8-103 "Vconn Source requests firmware update exchange with Cable Plug" shows an example sequence for a firmware update exchange between a VCONN Source and a Cable Plug.

Figure 8-103 "VCONN Source requests firmware update exchange with Cable Plug"

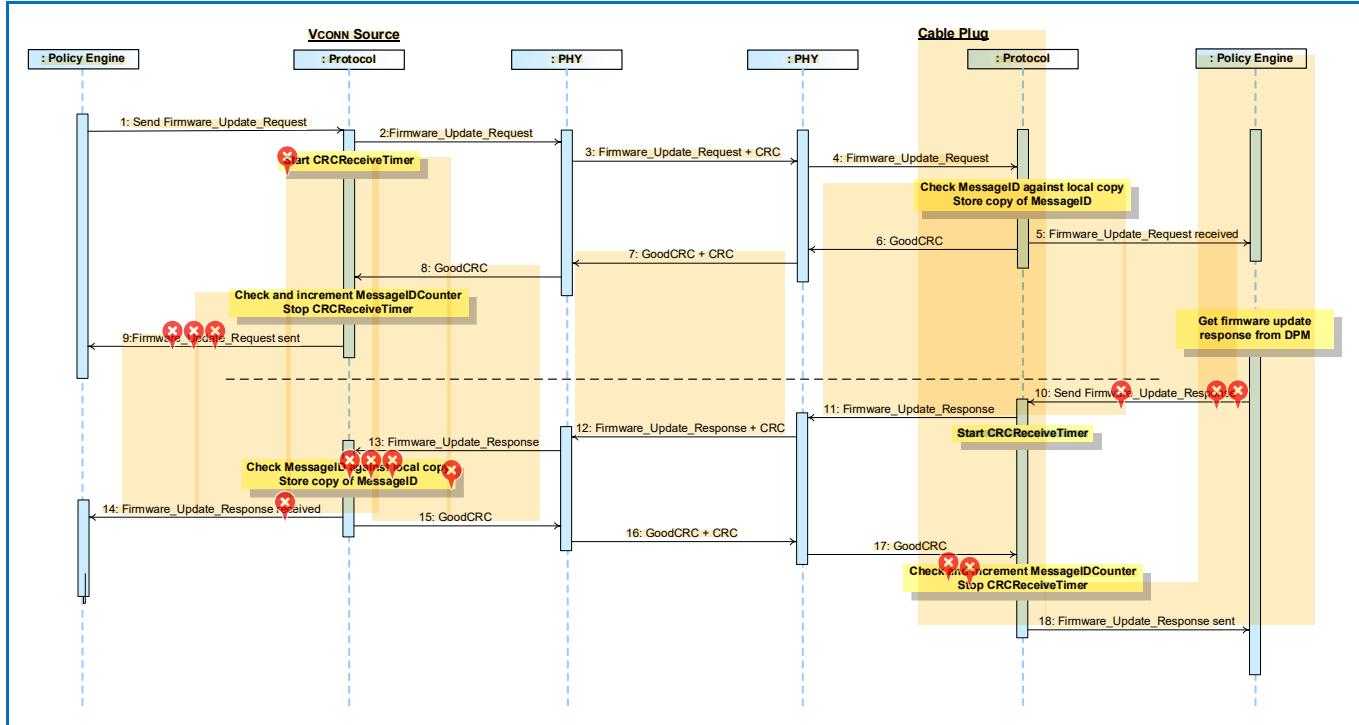


Table 8.131 “Steps for a Vconn Source requesting a firmware update exchange with a Cable Plug Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-103 “Vconn Source requests firmware update exchange with Cable Plug”** above.

Table 8.131 “Steps for a VCONN Source requesting a firmware update exchange with a Cable Plug Sequence”

Step	VCONN Source	Cable Plug
1	Policy Engine directs the Protocol Layer to send a Firmware_Update_Request Message using a payload supplied by the DPM.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Firmware_Update_Request Message. Starts CRCReceiveTimer .	Physical Layer receives the Firmware_Update_Request Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Firmware_Update_Request Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Firmware_Update_Request Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Firmware_Update_Request Message was successfully sent.	
10		Policy Engine requests the DPM for the response to the firmware update request which is provided. The Policy Engine tells the Protocol Layer to form a Firmware_Update_Response Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Firmware_Update_Response Message.	Physical Layer appends a CRC and sends the Firmware_Update_Response Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Firmware_Update_Response Message information to the Policy Engine that consumes it.	

14	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
15	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
16		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
17		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Firmware_Update_Response</i> Message was successfully sent.
The firmware update exchange is complete.		

8.3.2.15 Structured VDM

8.3.2.15.1 Discover Identity

8.3.2.15.1.1 Initiator to Responder Discover Identity (ACK)

Figure 8-104 “Initiator to Responder Discover Identity (ACK)” shows an example sequence between an Initiator and Responder, where both Port Partners are in an Explicit Contract and the Initiator discovers identity information from the Responder.

Figure 8-104 “Initiator to Responder Discover Identity (ACK)”

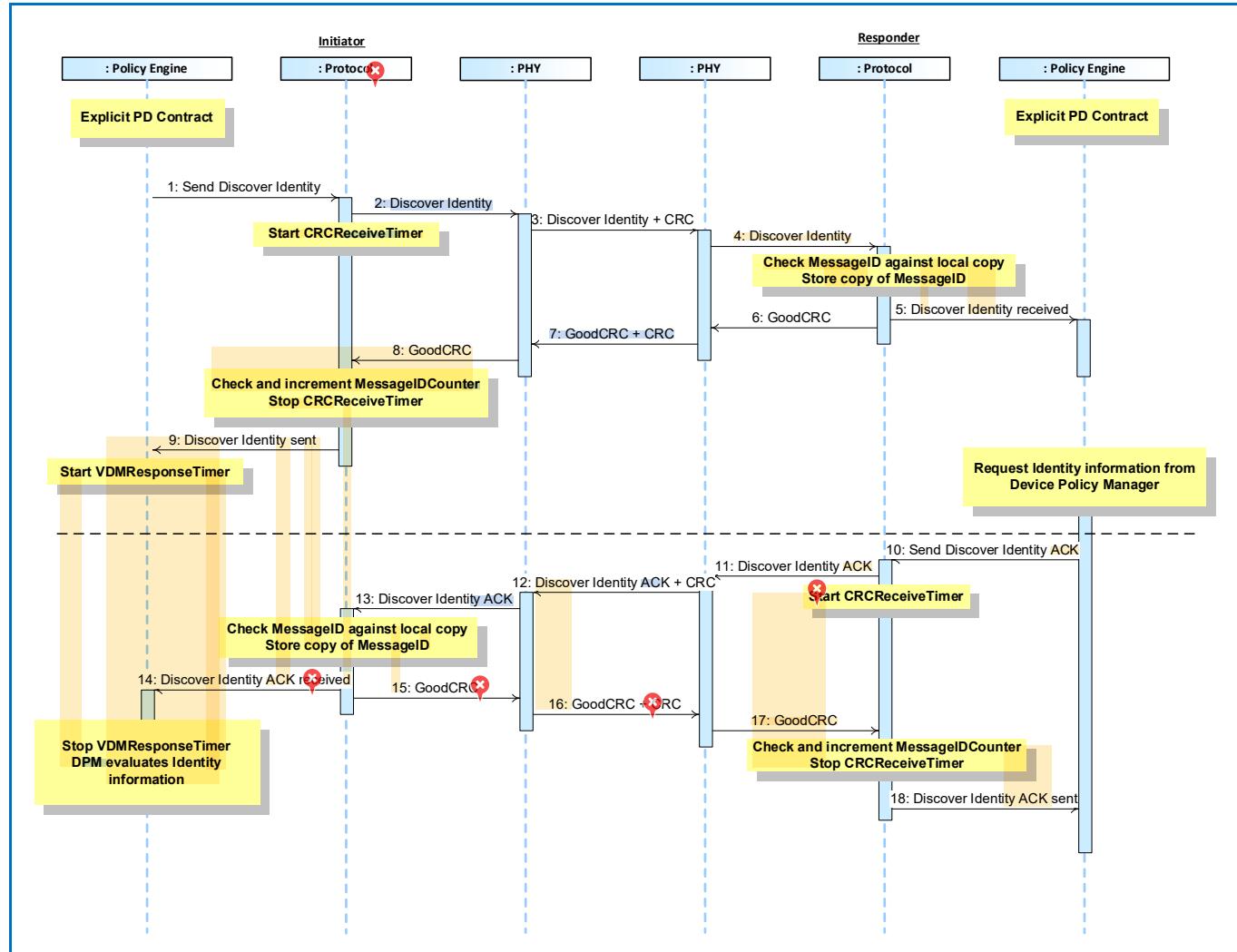


Table 8.132 "Steps for Initiator to UFP Discover Identity (ACK)" below provides a detailed explanation of what happens at each labeled step in **Figure 8-104 "Initiator to Responder Discover Identity (ACK)"** above.

Table 8.132 "Steps for Initiator to UFP Discover Identity (ACK)"

Step	Initiator	Responder
1	The Initiator has an Explicit Contract. The Policy Engine directs the Protocol Layer to send a Discover Identity Command request.	The Responder has an Explicit Contract.
2	Protocol Layer creates the Discover Identity Command request and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Discover Identity Command request. Starts CRCReceiveTimer .	Physical Layer receives the Discover Identity Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Discover Identity Command request to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover Identity Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Discover Identity Command request was successfully sent. Policy Engine starts the VDMResponseTimer .	
10		Policy Engine requests the identity information from the Device Policy Manager. The Policy Engine tells the Protocol Layer to form a Discover Identity Command ACK response.
11		Protocol Layer creates the Discover Identity Command ACK response and passes to Physical Layer.
12	Physical Layer receives the Discover Identity Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Discover Identity Command ACK response. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover Identity Command ACK response information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>VDMResponseTimer</i> and passed the Identity information to the Device Policy Manager for evaluation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Identity</i> Command ACK response was successfully sent.

8.3.2.15.1.2 Initiator to Responder Discover Identity (NAK)

Figure 8-105 “Initiator to Responder Discover Identity (NAK)” shows an example sequence between an Initiator and Responder, where both Port Partners are in an Explicit Contract and the Initiator attempts to discover identity information from the Responder but receives a NAK.

Figure 8-105 “Initiator to Responder Discover Identity (NAK)”

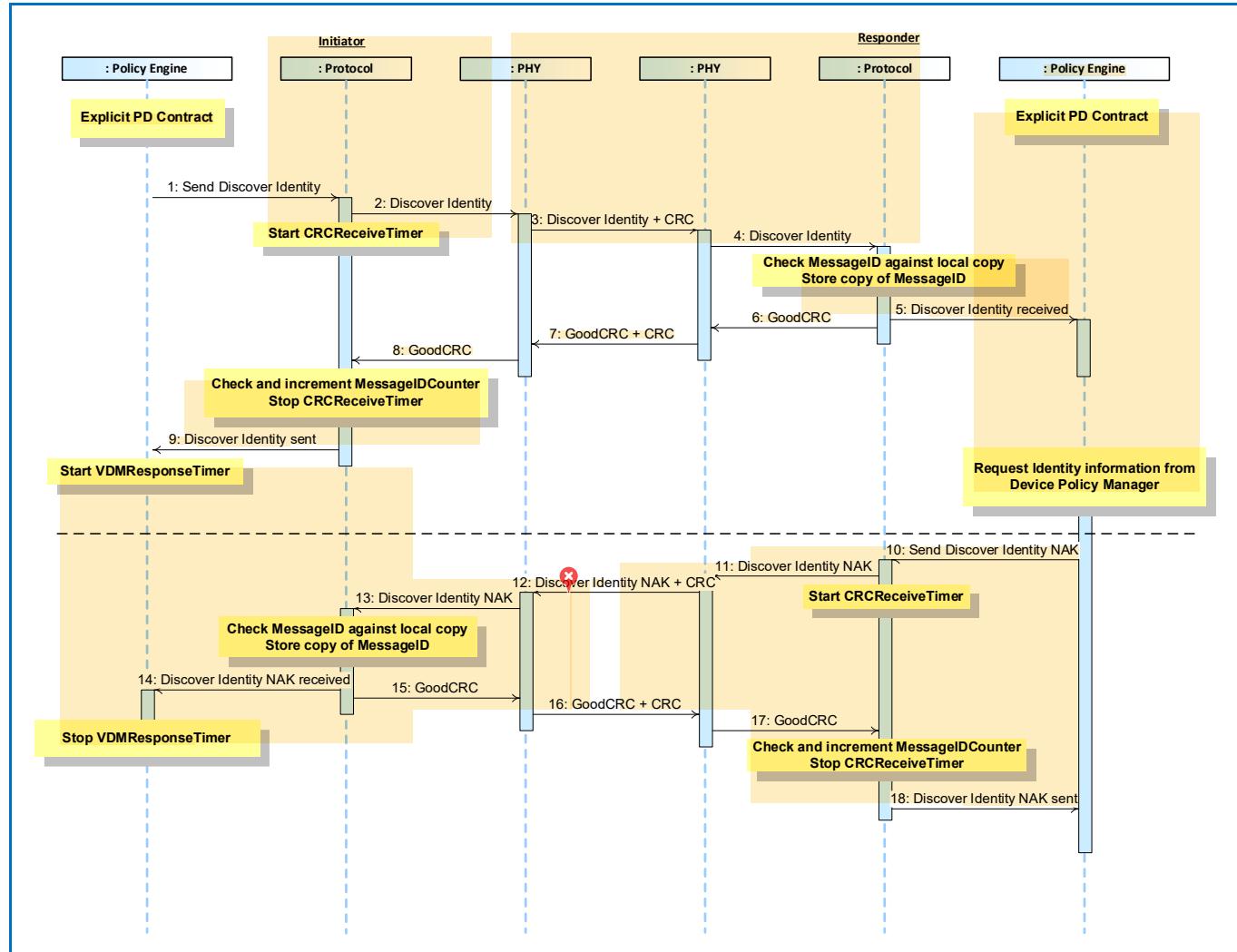


Table 8.133 “Steps for Initiator to UFP Discover Identity (NAK)” below provides a detailed explanation of what happens at each labeled step in **Figure 8-105 “Initiator to Responder Discover Identity (NAK)”** above.

Table 8.133 “Steps for Initiator to UFP Discover Identity (NAK)”

Step	Initiator	Responder
1	The Initiator has an Explicit Contract. The Policy Engine directs the Protocol Layer to send a Discover Identity Command request.	The Responder has an Explicit Contract.
2	Protocol Layer creates the Discover Identity Command request and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Discover Identity Command request. Starts CRCReceiveTimer .	Physical Layer receives the Discover Identity Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Discover Identity Command request to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover Identity Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Discover Identity Command request was successfully sent. Policy Engine starts the VDMResponseTimer .	
10		Policy Engine requests the identity information from the Device Policy Manager. The Policy Engine tells the Protocol Layer to form a Discover Identity Command NAK response.
11		Protocol Layer creates the Discover Identity Command NAK response and passes to Physical Layer.
12	Physical Layer receives the Discover Identity Command NAK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Discover Identity Command NAK response. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover Identity Command NAK response information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>VDMResponseTimer</i> and passed the Identity information to the Device Policy Manager for evaluation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Identity</i> Command NAK response was successfully sent.

8.3.2.15.1.3

Initiator to Responder Discover Identity (BUSY)

Figure 8-106 “Initiator to Responder Discover Identity (BUSY)” shows an example sequence between an Initiator and Responder, where both Port Partners are in an Explicit Contract and the Initiator attempts to discover identity information from the Responder but receives a BUSY.

Figure 8-106 “Initiator to Responder Discover Identity (BUSY)”

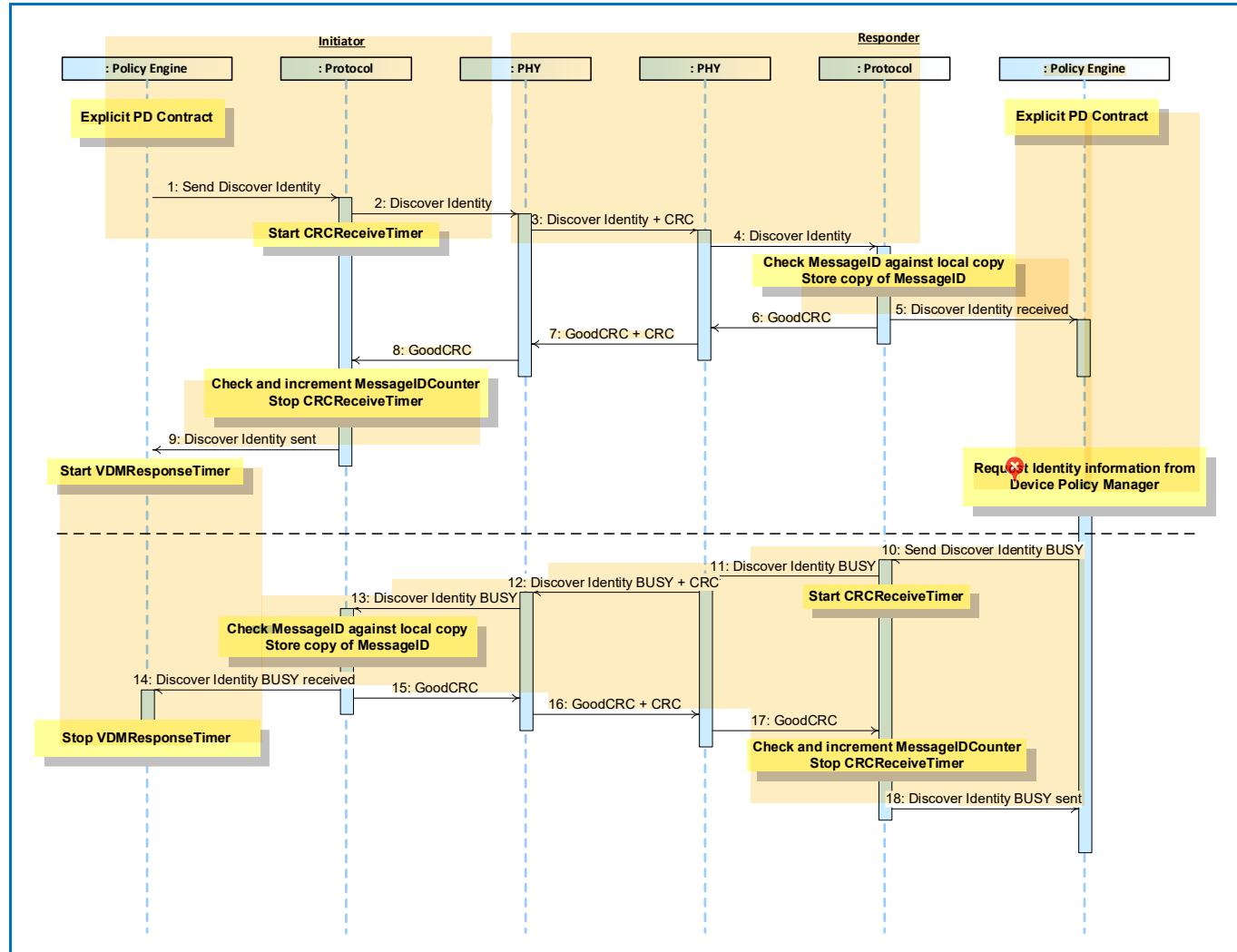


Table 8.134 "Steps for Initiator to UFP Discover Identity (BUSY)" below provides a detailed explanation of what happens at each labeled step in **Figure 8-106 "Initiator to Responder Discover Identity (BUSY)"** above.

📌 **Table 8.134 "Steps for Initiator to UFP Discover Identity (BUSY)"**

Step	Initiator	Responder
1	The Initiator has an Explicit Contract. The Policy Engine directs the Protocol Layer to send a Discover Identity Command request.	The Responder has an Explicit Contract.
2	Protocol Layer creates the Discover Identity Command request and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Discover Identity Command request. Starts CRCReceiveTimer .	Physical Layer receives the Discover Identity Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Discover Identity Command request to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover Identity Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Discover Identity Command request was successfully sent. Policy Engine starts the VDMResponseTimer .	
10		Policy Engine requests the identity information from the Device Policy Manager. The Policy Engine tells the Protocol Layer to form a Discover Identity Command BUSY response.
11		Protocol Layer creates the Discover Identity Command BUSY response and passes to Physical Layer.
12	Physical Layer receives the Discover Identity Command BUSY response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Discover Identity Command BUSY response. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover Identity Command BUSY response information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>VDMResponseTimer</i> and passed the Identity information to the Device Policy Manager for evaluation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Identity</i> Command NAK response was successfully sent.

8.3.2.15.2 Discover SVIDs

8.3.2.15.2.1 Initiator to Responder Discover SVIDs (ACK)

Figure 8-107 “Initiator to Responder Discover SVIDs (ACK)” shows an example sequence between an Initiator and Responder, where both Port Partners are in an Explicit Contract and the Initiator discovers SVID information from the Responder.

Figure 8-107 “Initiator to Responder Discover SVIDs (ACK)”

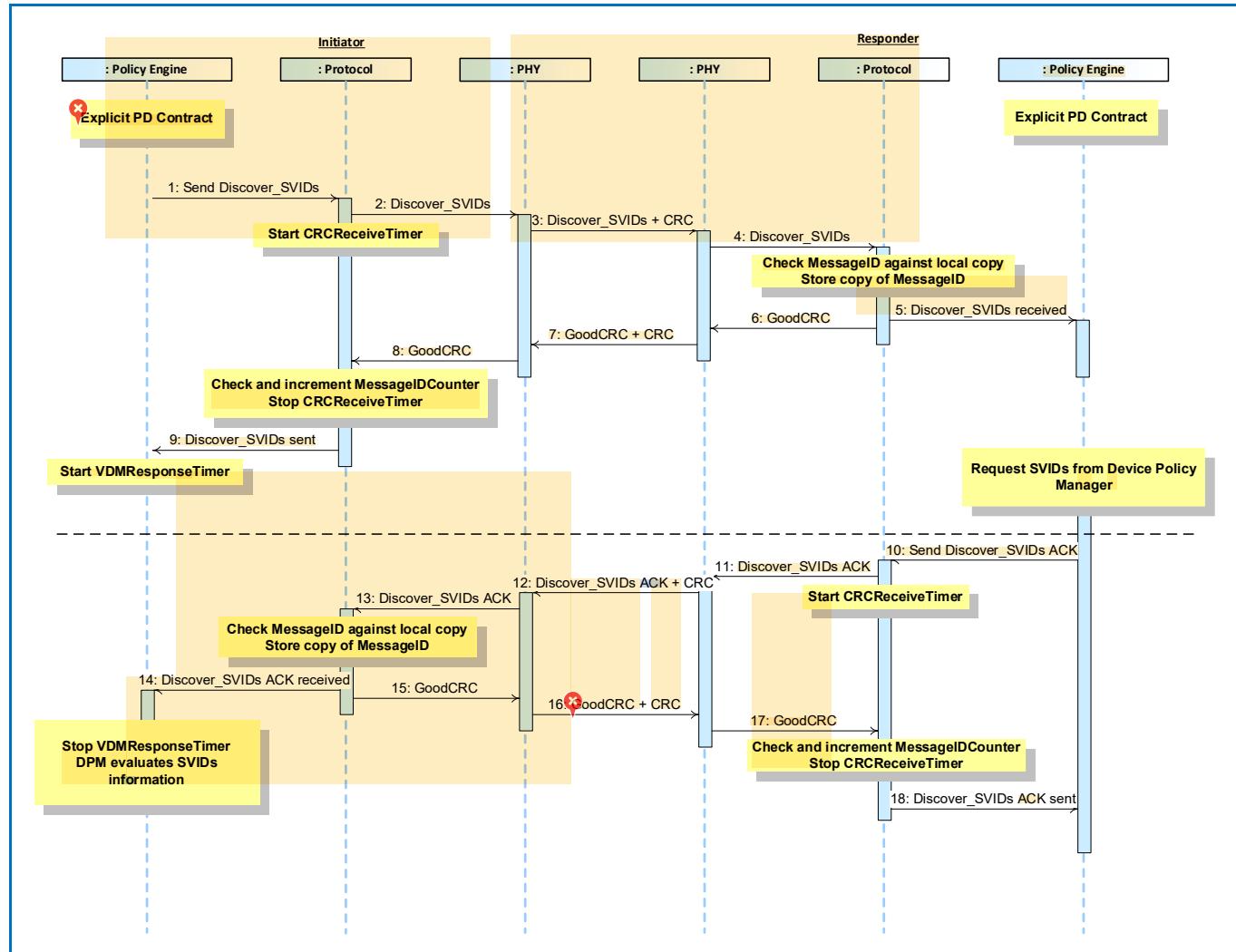


Table 8.135 "Steps for DFP to UFP Discover SVIDs (ACK)" below provides a detailed explanation of what happens at each labeled step in **Figure 8-107 "Initiator to Responder Discover SVIDs (ACK)"** above.

✖ **Table 8.135 "Steps for DFP to UFP Discover SVIDs (ACK)"**

Step	Initiator	Responder
1	The Initiator has an Explicit Contract. The Policy Engine directs the Protocol Layer to send a Discover SVIDs Command request.	The Responder has an Explicit Contract.
2	Protocol Layer creates the Discover SVIDs Command request and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Discover SVIDs Command request. Starts CRCReceiveTimer .	Physical Layer receives the Discover SVIDs Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Discover SVIDs Command request to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover SVIDs Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Discover SVIDs Command request was successfully sent. Policy Engine starts the VDMResponseTimer .	
10		Policy Engine requests the identity information from the Device Policy Manager. The Policy Engine tells the Protocol Layer to form a Discover SVIDs Command ACK response.
11		Protocol Layer creates the Discover SVIDs Command ACK response and passes to Physical Layer.
12	Physical Layer receives the Discover SVIDs Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Discover SVIDs Command ACK response. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover SVIDs Command ACK response information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>VDMResponseTimer</i> and passed the Identity information to the Device Policy Manager for evaluation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover SVIDs</i> Command ACK response was successfully sent.

8.3.2.15.2.2

Initiator to Responder Discover SVIDs (NAK)

Figure 8-108 “Initiator to Responder Discover SVIDs (NAK)” shows an example sequence between an Initiator and Responder, where both Port Partners are in an Explicit Contract and the Initiator attempts to discover SVID information from the Responder but receives a NAK.

Figure 8-108 “Initiator to Responder Discover SVIDs (NAK)”

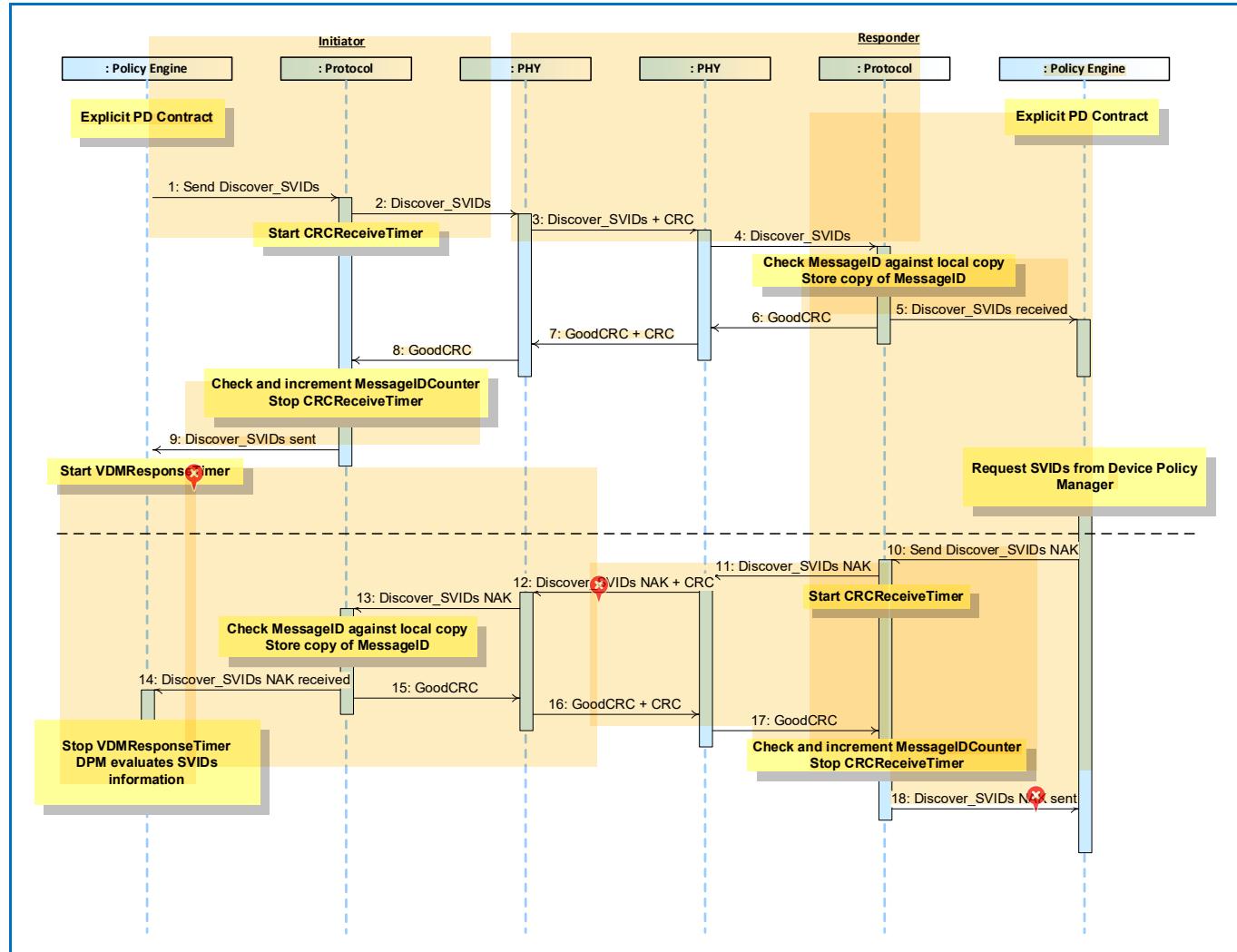


Table 8.136 “Steps for DFP to UFP Discover SVIDs (NAK)” below provides a detailed explanation of what happens at each labeled step in **Figure 8-108 “Initiator to Responder Discover SVIDs (NAK)”** above.

Table 8.136 “Steps for DFP to UFP Discover SVIDs (NAK)”

Step	Initiator	Responder
1	The Initiator has an Explicit Contract. The Policy Engine directs the Protocol Layer to send a Discover SVIDs Command request.	The Responder has an Explicit Contract.
2	Protocol Layer creates the Discover SVIDs Command request and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Discover SVIDs Command request. Starts CRCReceiveTimer .	Physical Layer receives the Discover SVIDs Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Discover SVIDs Command request to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover SVIDs Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Discover SVIDs Command request was successfully sent. Policy Engine starts the VDMResponseTimer .	
10		Policy Engine requests the identity information from the Device Policy Manager. The Policy Engine tells the Protocol Layer to form a Discover SVIDs Command NAK response.
11		Protocol Layer creates the Discover SVIDs Command NAK response and passes to Physical Layer.
12	Physical Layer receives the Discover SVIDs Command NAK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Discover SVIDs Command NAK response. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover SVIDs Command NAK response information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>VDMResponseTimer</i> and passed the Identity information to the Device Policy Manager for evaluation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover SVIDs</i> Command NAK response was successfully sent.

8.3.2.15.2.3

Initiator to Responder Discover SVIDs (BUSY)

Figure 8-109 “Initiator to Responder Discover SVIDs (BUSY)” shows an example sequence between an Initiator and Responder, where both Port Partners are in an Explicit Contract and the Initiator attempts to discover SVID information from the Responder but receives a BUSY.

Figure 8-109 “Initiator to Responder Discover SVIDs (BUSY)”

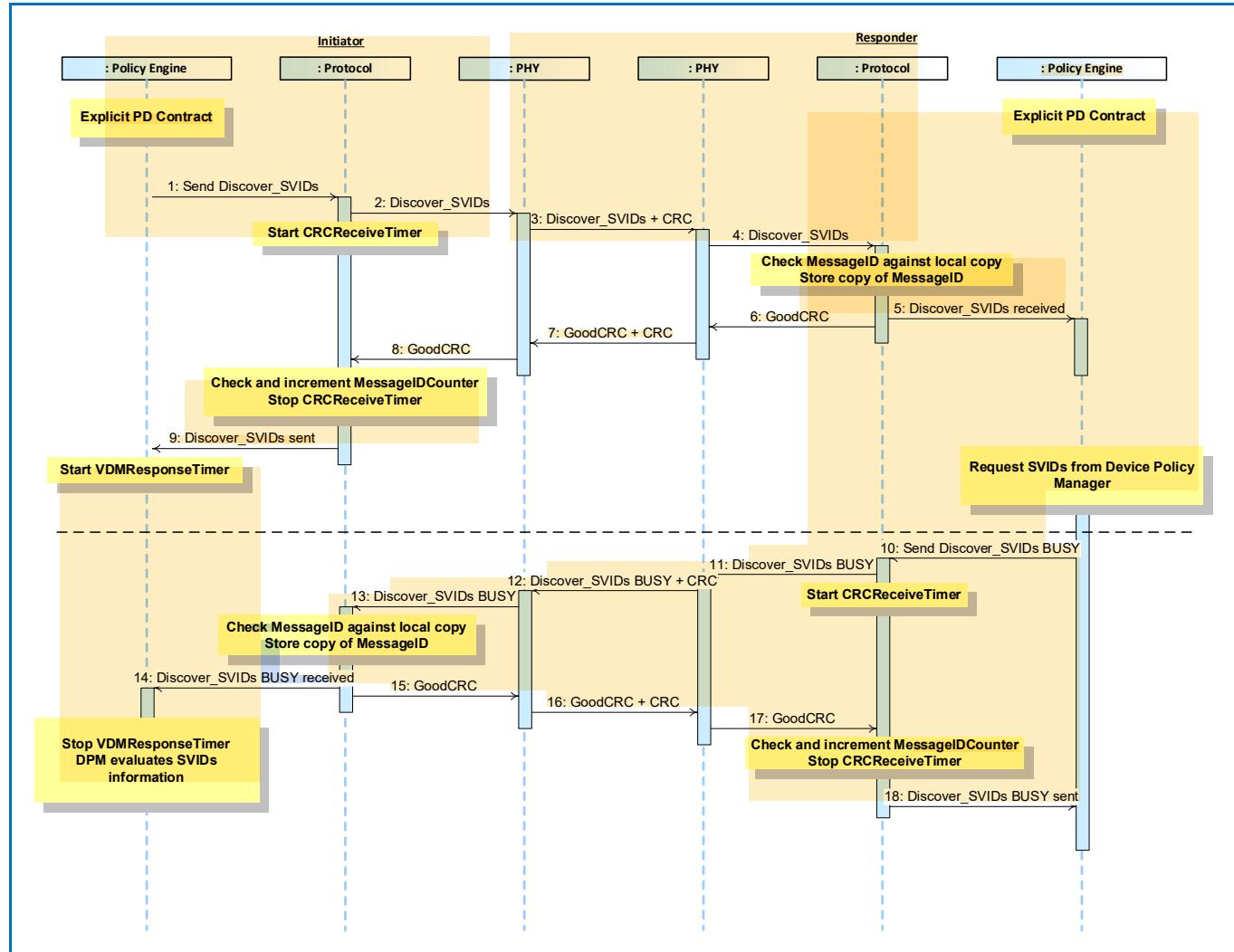


Table 8.137 "Steps for DFP to UFP Discover SVIDs (BUSY)" below provides a detailed explanation of what happens at each labeled step in **Figure 8-109 "Initiator to Responder Discover SVIDs (BUSY)"** above.

Table 8.137 "Steps for DFP to UFP Discover SVIDs (BUSY)"

Step	Initiator	Responder
1	The Initiator has an Explicit Contract. The Policy Engine directs the Protocol Layer to send a Discover SVIDs Command request.	The Responder has an Explicit Contract.
2	Protocol Layer creates the Discover SVIDs Command request and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Discover SVIDs Command request. Starts CRCReceiveTimer .	Physical Layer receives the Discover SVIDs Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Discover SVIDs Command request to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover SVIDs Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Discover SVIDs Command request was successfully sent. Policy Engine starts the VDMResponseTimer .	
10		Policy Engine requests the identity information from the Device Policy Manager. The Policy Engine tells the Protocol Layer to form a Discover SVIDs Command BUSY response.
11		Protocol Layer creates the Discover SVIDs Command BUSY response and passes to Physical Layer.
12	Physical Layer receives the Discover SVIDs Command BUSY response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Discover SVIDs Command BUSY response. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover SVIDs Command BUSY response information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>VDMResponseTimer</i> and passed the Identity information to the Device Policy Manager for evaluation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover SVIDs</i> Command BUSY response was successfully sent.

8.3.2.15.3 Discover Modes

8.3.2.15.3.1 Initiator to Responder Discover Modes (ACK)

Figure 8-110 “Initiator to Responder Discover Modes (ACK)” shows an example sequence between an Initiator and Responder, where both Port Partners are in an Explicit Contract and the Initiator discovers Mode information from the Responder.

Figure 8-110 “Initiator to Responder Discover Modes (ACK)”

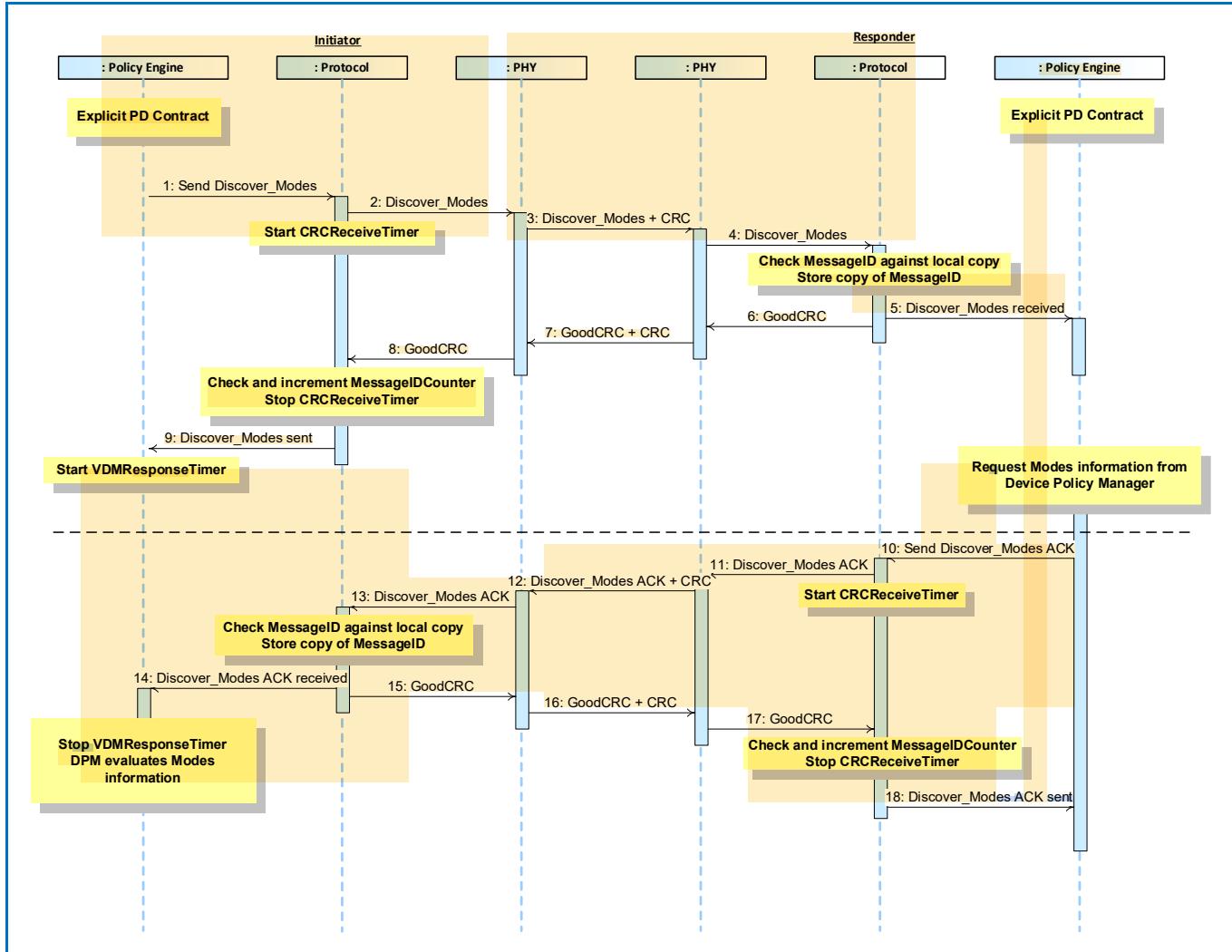


Table 8.138 "Steps for DFP to UFP Discover Modes (ACK)" below provides a detailed explanation of what happens at each labeled step in **Figure 8-110 "Initiator to Responder Discover Modes (ACK)"**.

Table 8.138 "Steps for DFP to UFP Discover Modes (ACK)"

Step	DFP	UFP
1	The DFP has an Explicit Contract. The Policy Engine directs the Protocol Layer to send a Discover Modes Command request.	The UFP has an Explicit Contract.
2	Protocol Layer creates the Discover Modes Command request and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Discover Modes Command request. Starts CRCReceiveTimer .	Physical Layer receives the Discover Modes Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Discover Modes Command request to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover Modes Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Discover Modes Command request was successfully sent. Policy Engine starts the VDMResponseTimer .	
10		Policy Engine requests the identity information from the Device Policy Manager. The Policy Engine tells the Protocol Layer to form a Discover Modes Command ACK response.
11		Protocol Layer creates the Discover Modes Command ACK response and passes to Physical Layer.
12	Physical Layer receives the Discover Modes Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Discover Modes Command ACK response. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover Modes Command ACK response information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>VDMResponseTimer</i> and passed the Identity information to the Device Policy Manager for evaluation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Modes</i> Command ACK response was successfully sent.

8.3.2.15.3.2

Initiator to Responder Discover Modes (NAK)

Figure 8-111 “Initiator to Responder Discover Modes (NAK)” shows an example sequence between an Initiator and Responder, where both Port Partners are in an Explicit Contract and the Initiator attempts to discover Mode information from the Responder but receives a NAK.

Figure 8-111 “Initiator to Responder Discover Modes (NAK)”

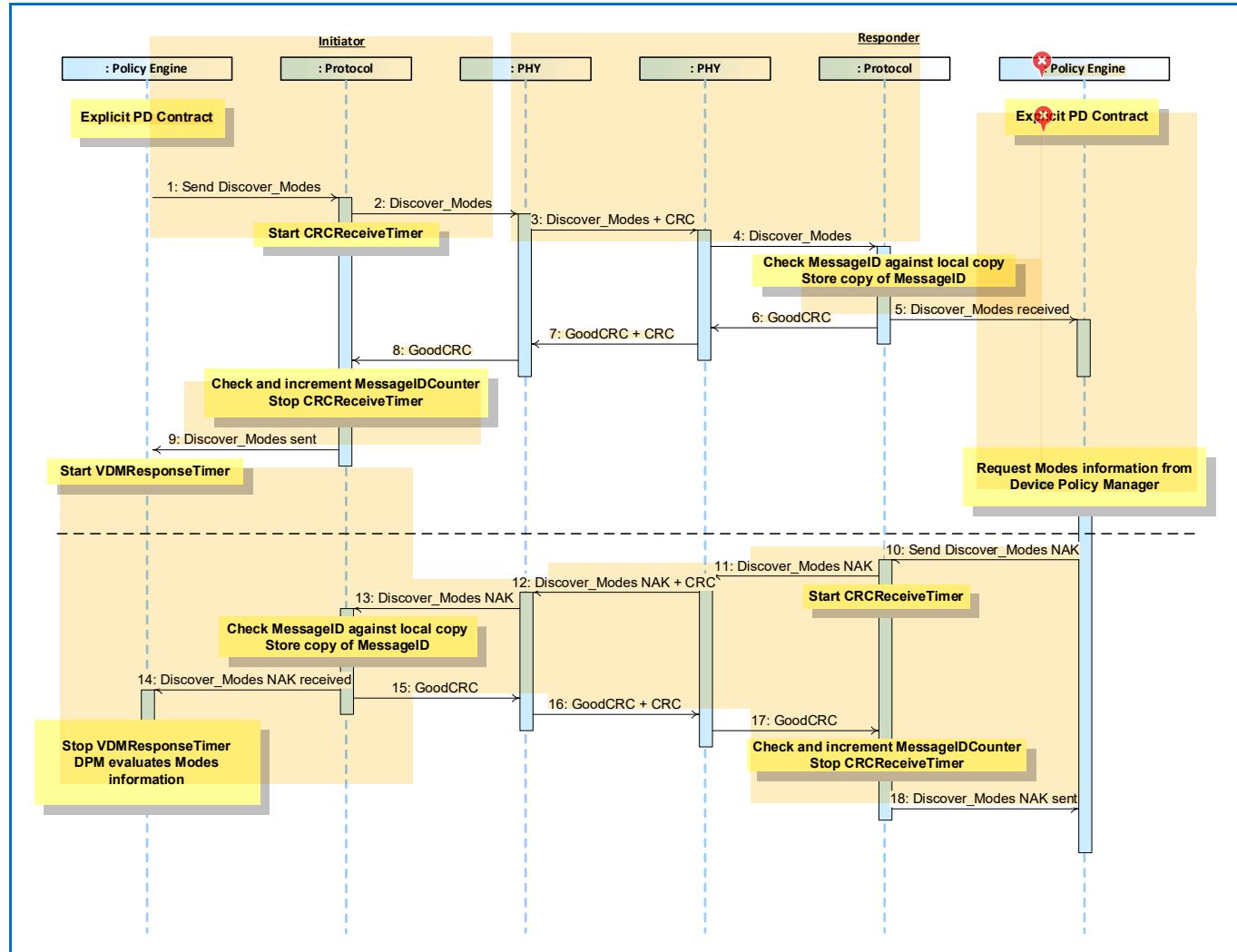


Table 8.139 "Steps for DFP to UFP Discover Modes (NAK)" below provides a detailed explanation of what happens at each labeled step in **Figure 8-111 "Initiator to Responder Discover Modes (NAK)"**.

Table 8.139 "Steps for DFP to UFP Discover Modes (NAK)"

Step	DFP	UFP
1	The DFP has an Explicit Contract. The Policy Engine directs the Protocol Layer to send a Discover Modes Command request.	The UFP has an Explicit Contract.
2	Protocol Layer creates the Discover Modes Command request and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Discover Modes Command request. Starts CRCReceiveTimer .	Physical Layer receives the Discover Modes Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Discover Modes Command request to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover Modes Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Discover Modes Command request was successfully sent. Policy Engine starts the VDMResponseTimer .	
10		Policy Engine requests the identity information from the Device Policy Manager. The Policy Engine tells the Protocol Layer to form a Discover Modes Command NAK response.
11		Protocol Layer creates the Discover Modes Command NAK response and passes to Physical Layer.
12	Physical Layer receives the Discover Modes Command NAK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Discover Modes Command NAK response. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover Modes Command NAK response information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>VDMResponseTimer</i> and passed the Identity information to the Device Policy Manager for evaluation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Modes</i> Command NAK response was successfully sent.

8.3.2.15.3.3

Initiator to Responder Discover Modes (BUSY)

Figure 8-112 “Initiator to Responder Discover Modes (BUSY)” shows an example sequence between an Initiator and Responder, where both Port Partners are in an Explicit Contract and the Initiator attempts to discover Mode information from the Responder but receives a BUSY.

Figure 8-112 “Initiator to Responder Discover Modes (BUSY)”

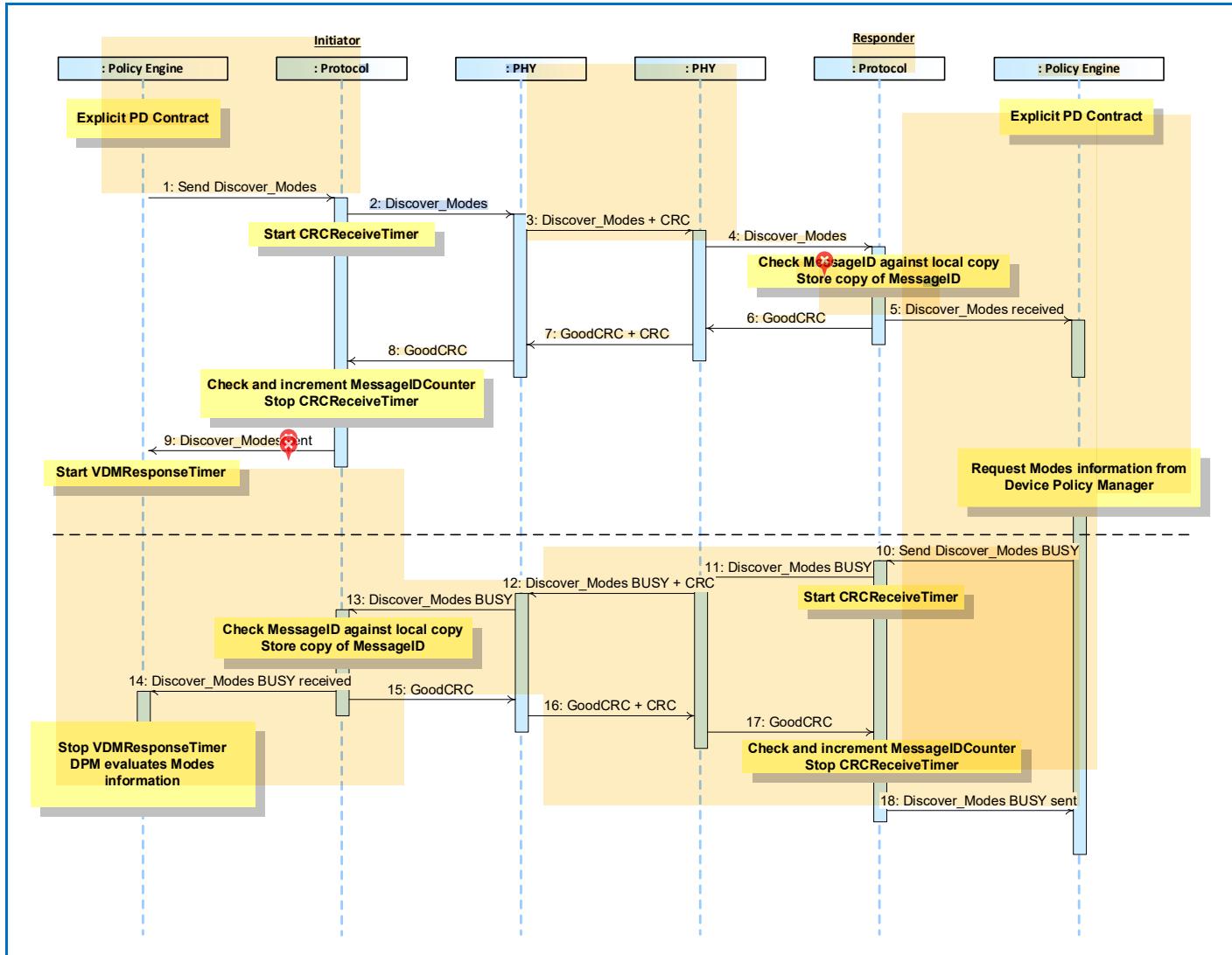


Table 8.140 “Steps for DFP to UFP Discover Modes (BUSY)” below provides a detailed explanation of what happens at each labeled step in **Figure 8-112 “Initiator to Responder Discover Modes (BUSY)”**.

Table 8.140 “Steps for DFP to UFP Discover Modes (BUSY)”

Step	DFP	UFP
1	The DFP has an Explicit Contract. The Policy Engine directs the Protocol Layer to send a Discover Modes Command request.	The UFP has an Explicit Contract.
2	Protocol Layer creates the Discover Modes Command request and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Discover Modes Command request. Starts CRCReceiveTimer .	Physical Layer receives the Discover Modes Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Discover Modes Command request to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover Modes Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Discover Modes Command request was successfully sent. Policy Engine starts the VDMResponseTimer .	
10		Policy Engine requests the identity information from the Device Policy Manager. The Policy Engine tells the Protocol Layer to form a Discover Modes Command NAK response.
11		Protocol Layer creates the Discover Modes Command NAK response and passes to Physical Layer.
12	Physical Layer receives the Discover Modes Command NAK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Discover Modes Command NAK response. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Discover Modes Command NAK response information to the Policy Engine that consumes it.	

14	The Policy Engine stops the <i>VDMResponseTimer</i> and passed the Identity information to the Device Policy Manager for evaluation.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Discover Modes</i> Command NAK response was successfully sent.

8.3.2.15.4 Enter/Exit Mode

8.3.2.15.4.1 DFP to UFP Enter Mode

Figure 8-113 “DFP to UFP Enter Mode” shows an example sequence between a DFP and a UFP that occurs after the DFP has discovered supported SVIDs and Modes at which point it selects and enters a Mode.

Figure 8-113 “DFP to UFP Enter Mode”

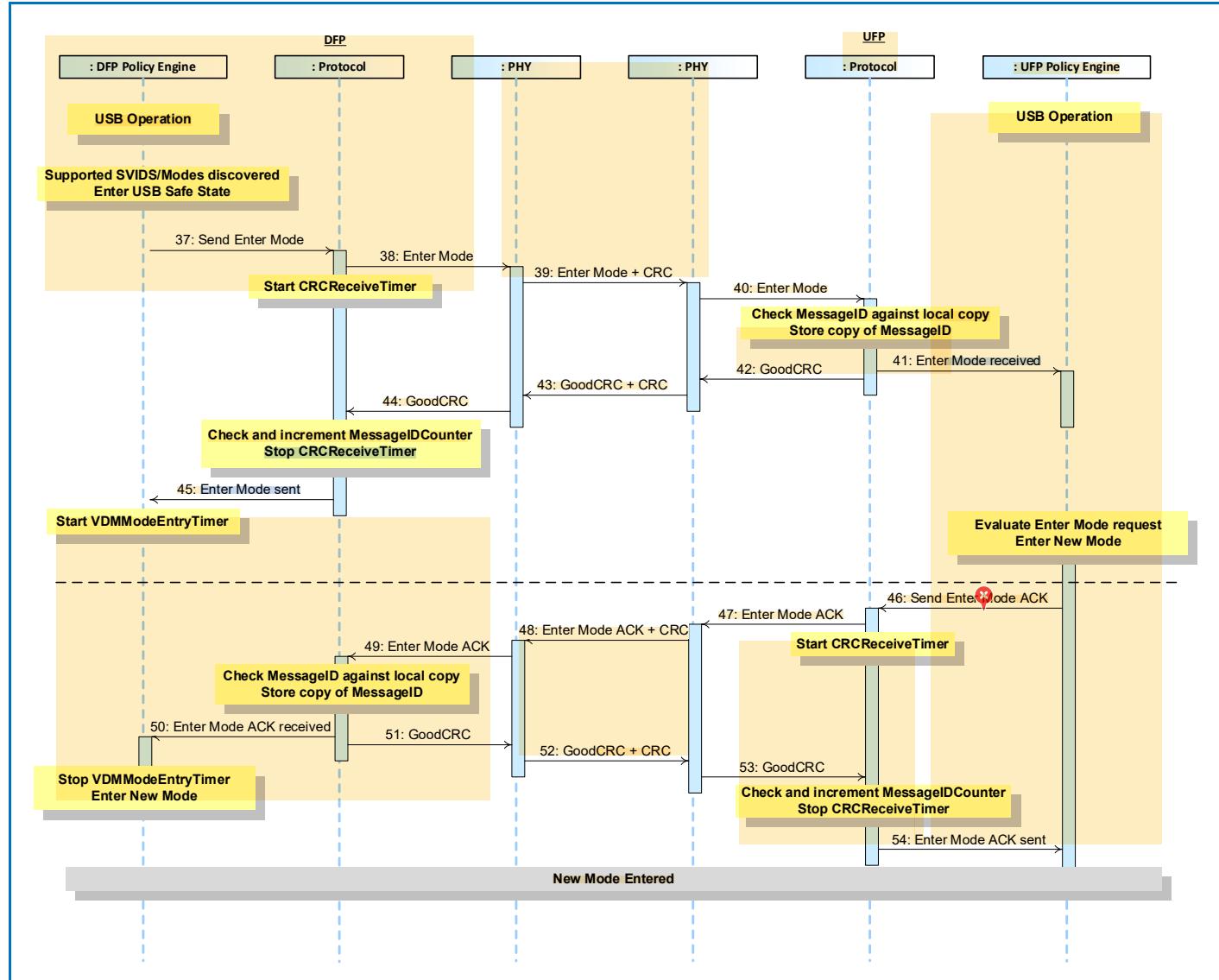


Table 8.141 “Steps for DFP to UFP Enter Mode” below provides a detailed explanation of what happens at each labeled step in **Figure 8-113 “DFP to UFP Enter Mode”** above.

Table 8.141 “Steps for DFP to UFP Enter Mode”

Step	DFP	UFP
1	<p>The DFP has an Explicit Contract</p> <p>The DFP has discovered the supported SVIDS using the <i>Discover SVIDs</i> Command request and the supported Modes using the <i>Discover Modes</i> Command request</p> <p>The DFP goes to USB Safe State. The Device Policy Manager requests the Policy Engine to enter a Mode.</p> <p>The Policy Engine directs the Protocol Layer to send an <i>Enter Mode</i> Command request.</p>	The UFP has an Explicit Contract.
2	Protocol Layer creates the <i>Enter Mode</i> Command request and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the <i>Enter Mode</i> Command request. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the <i>Enter Mode</i> Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Enter Mode</i> Command request to the Protocol Layer.
5		<p>Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value.</p> <p>The Protocol Layer forwards the received <i>Enter Mode</i> Command request information to the Policy Engine that consumes it.</p>
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	<p>Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i>.</p> <p>Protocol Layer informs the Policy Engine that the <i>Enter Mode</i> Command request was successfully sent.</p> <p>Policy Engine starts the <i>VDMModeEntryTimer</i>.</p>	
10		Policy Engine requests the Device Policy Manager to enter the new Mode. The Policy Engine tells the Protocol Layer to form an <i>Enter Mode</i> Command ACK response.
11		Protocol Layer creates the <i>Enter Mode</i> Command ACK response and passes to Physical Layer.
12	Physical Layer receives the <i>Enter Mode</i> Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the <i>Enter Mode</i> Command ACK response. Starts <i>CRCReceiveTimer</i> .

13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Enter Mode</i> Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>VDMModeEntryTimer</i> and requests the Device Policy Manager to enter the new Mode.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Enter Mode</i> Command ACK response was successfully sent.
DFP and UFP are operating in the new Mode		

8.3.2.15.4.2 DFP to UFP Exit Mode

Figure 8-114 “DFP to UFP Exit Mode” shows an example sequence between a DFP and a UFP, where the DFP commands the UFP to exit the only *Active Mode*.

Figure 8-114 “DFP to UFP Exit Mode”

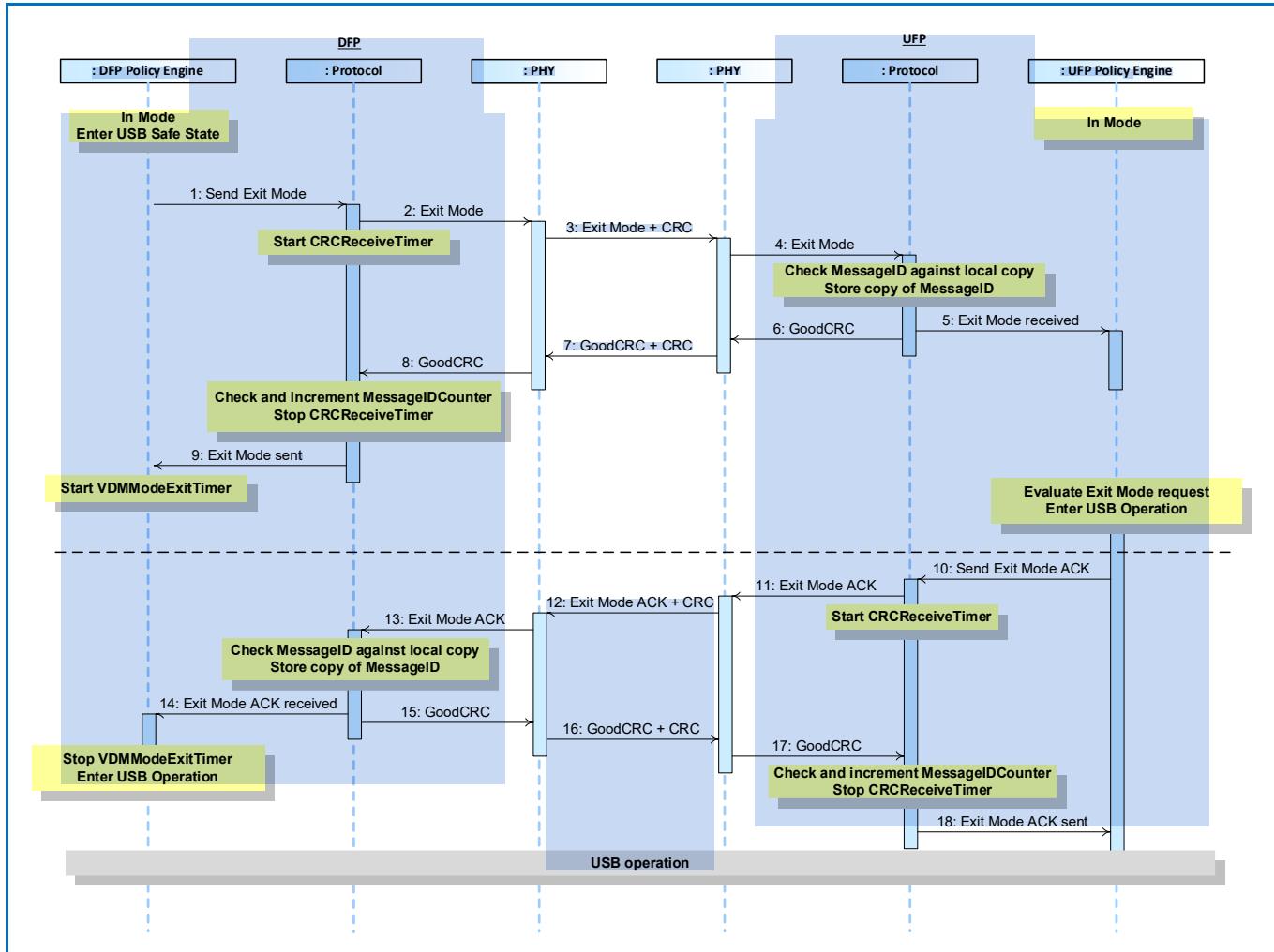


Table 8.142 “Steps for DFP to UFP Exit Mode” below provides a detailed explanation of what happens at each labeled step in **Figure 8-114 “DFP to UFP Exit Mode”** above.

Table 8.142 “Steps for DFP to UFP Exit Mode”

Step	DFP	UFP
1	The DFP is in a Mode and then enters USB Safe State. The Policy Engine directs the Protocol Layer to send an Exit Mode Command request.	The UFP is in a Mode.
2	Protocol Layer creates the Exit Mode Command request and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Exit Mode Command request. Starts CRCReceiveTimer .	Physical Layer receives the Exit Mode Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Exit Mode Command request to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Exit Mode Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Exit Mode Command request was successfully sent. Policy Engine starts the VDMModeExitTimer .	
10		Policy Engine requests the Device Policy Manager to enter USB operation. The Policy Engine tells the Protocol Layer to form an Exit Mode Command ACK response.
11		Protocol Layer creates the Exit Mode Command ACK response and passes to Physical Layer.
12	Physical Layer receives the Exit Mode Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Exit Mode Command ACK response. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Exit Mode Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the VDMModeExitTimer and requests the Device Policy Manager to enter USB Operation.	

15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
18		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Exit Mode Command ACK response was successfully sent.
Both DFP and UFP are in USB Operation		

8.3.2.15.4.3 DFP to Cable Plug Enter Mode

Figure 8-115 “DFP to Cable Plug Enter Mode” shows an example sequence between a DFP and a Cable Plug that occurs after the DFP has discovered supported SVIDs and Modes at which point it selects and enters a Mode.

Figure 8-115 “DFP to Cable Plug Enter Mode”

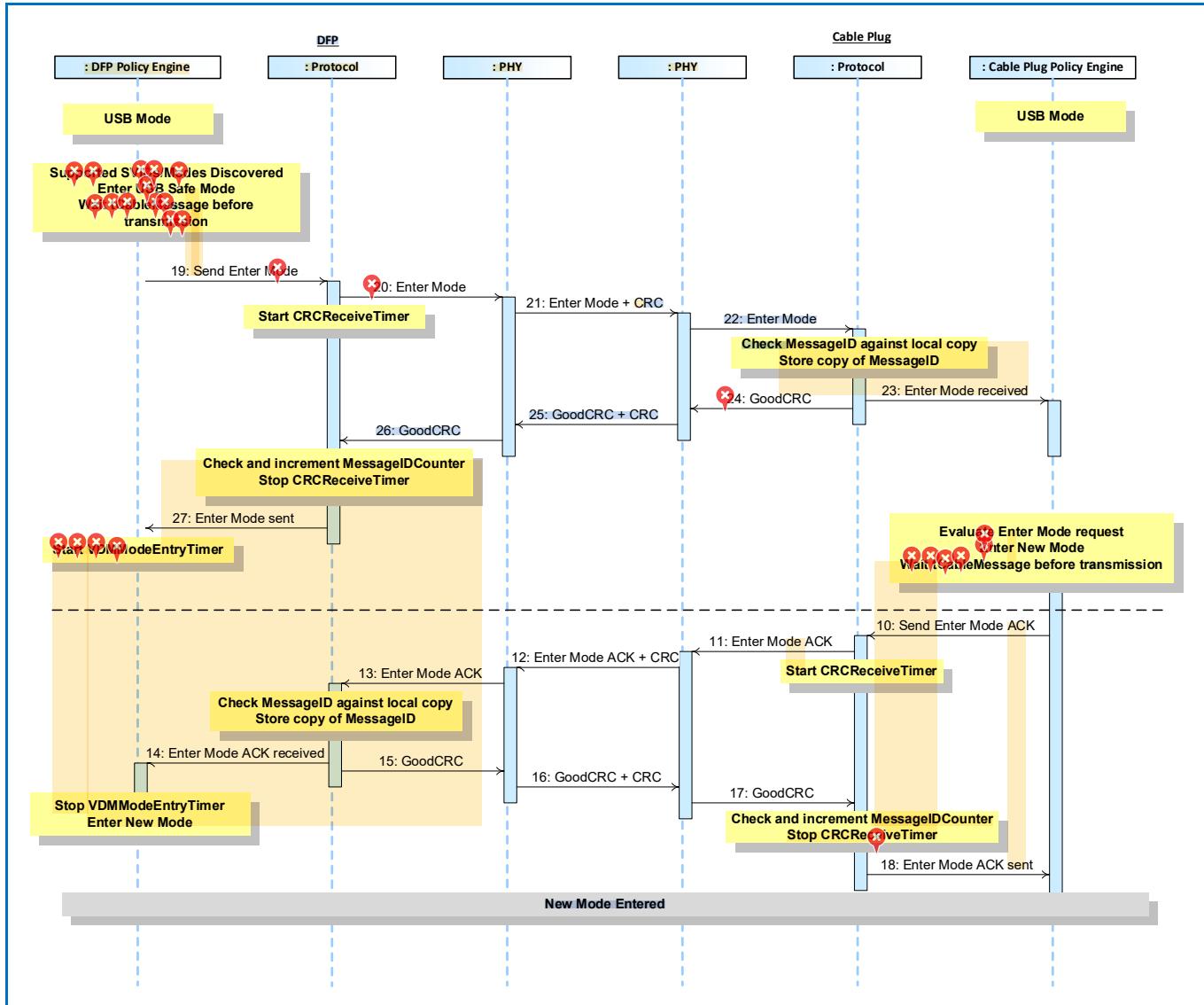


Table 8.143 "Steps for DFP to Cable Plug Enter Mode" below provides a detailed explanation of what happens at each labeled step in **Figure 8-115 "DFP to Cable Plug Enter Mode"** above.

Table 8.143 "Steps for DFP to Cable Plug Enter Mode"

Step	DFP	Cable Plug
1	The DFP has an Explicit Contract The DFP has discovered the supported SVIDs using the Discover SVIDs Command request and the supported Modes using the Discover Modes Command request The DFP goes to USB Safe State. The Device Policy Manager requests the Policy Engine to enter a Mode. tCableMessage after the last GoodCRC Message was sent the Policy Engine directs the Protocol Layer to send an Enter Mode Command request.	
2	Protocol Layer creates the Enter Mode Command request and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Enter Mode Command request. Starts CRCReceiveTimer .	Physical Layer receives the Enter Mode Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Enter Mode Command request to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Enter Mode Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Enter Mode Command request was successfully sent. Policy Engine starts the VDMModeEntryTimer .	
10		Policy Engine requests the Device Policy Manager to enter the new Mode. tCableMessage after the GoodCRC Message was sent the Policy Engine tells the Protocol Layer to form an Enter Mode Command ACK response.
11		Protocol Layer creates the Enter Mode Command ACK response and passes to Physical Layer.
12	Physical Layer receives the Enter Mode Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Enter Mode Command ACK response. Starts CRCReceiveTimer .

13	Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Enter Mode</i> Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the <i>VDMModeEntryTimer</i> and requests the Device Policy Manager to enter the new Mode.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Enter Mode</i> Command ACK response was successfully sent.
DFP and Cable Plug are operating in the new Mode		

8.3.2.15.4.4 DFP to Cable Plug Exit Mode

Figure 8-116 “DFP to Cable Plug Exit Mode” shows an example sequence between a USB Type-C® DFP and a Cable Plug, where the DFP commands the Cable Plug to exit an *Active Mode*.

Figure 8-116 “DFP to Cable Plug Exit Mode”

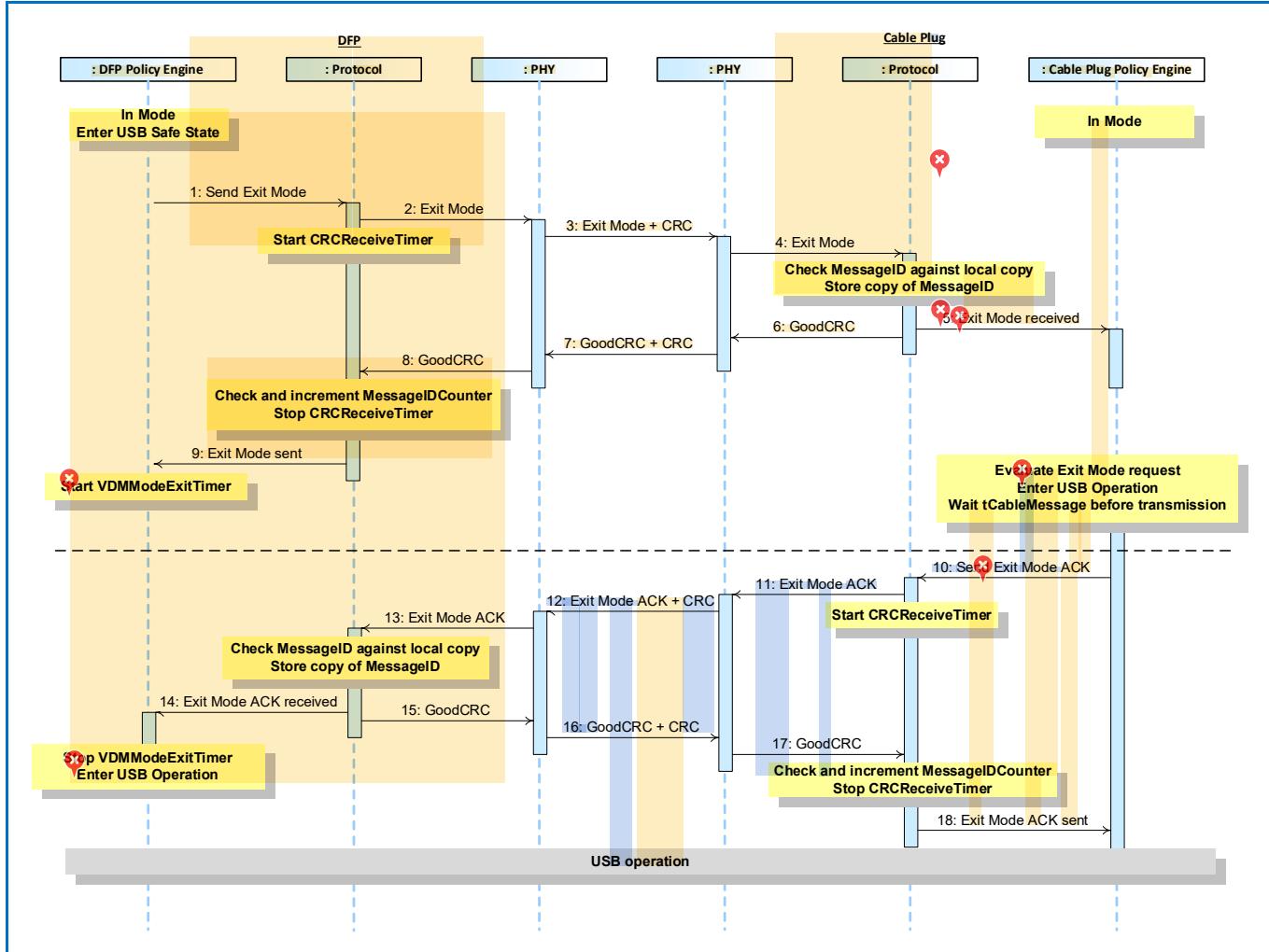


Table 8.144 "Steps for DFP to Cable Plug Exit Mode" below provides a detailed explanation of what happens at each labeled step in **Figure 8-116 "DFP to Cable Plug Exit Mode"** above.

Table 8.144 "Steps for DFP to Cable Plug Exit Mode"

Step	DFP	Cable Plug
1	The DFP is in a Mode and then enters USB Safe State. The Policy Engine directs the Protocol Layer to send an Exit Mode Command request.	The Cable Plug is in a Mode.
2	Protocol Layer creates the Exit Mode Command request and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Exit Mode Command request. Starts CRCReceiveTimer .	Physical Layer receives the Exit Mode Command request and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Exit Mode Command request to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Exit Mode Command request information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Exit Mode Command request was successfully sent. Policy Engine starts the VDMModeExitTimer .	
10		Policy Engine requests the Device Policy Manager to enter USB operation. tCableMessage after the GoodCRC Message was sent the Policy Engine tells the Protocol Layer to form an Exit Mode Command ACK response.
11		Protocol Layer creates the Exit Mode Command ACK response and passes to Physical Layer.
12	Physical Layer receives the Exit Mode Command ACK response and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Exit Mode Command ACK response. Starts *CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Exit Mode Command ACK response information to the Policy Engine that consumes it.	
14	The Policy Engine stops the VDMModeExitTimer and requests the Device Policy Manager to enter USB Operation.	

15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the  <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Exit Mode</i> Command ACK response was successfully sent.
 Both DFP and Cable Plug are in USB Operation		

8.3.2.15.5

Initiator to Responder Attention

Figure 8-117 “Initiator to Responder Attention” shows an example sequence between an Initiator and a Responder, where the Initiator requests attention from the Responder.

Figure 8-117 “Initiator to Responder Attention”

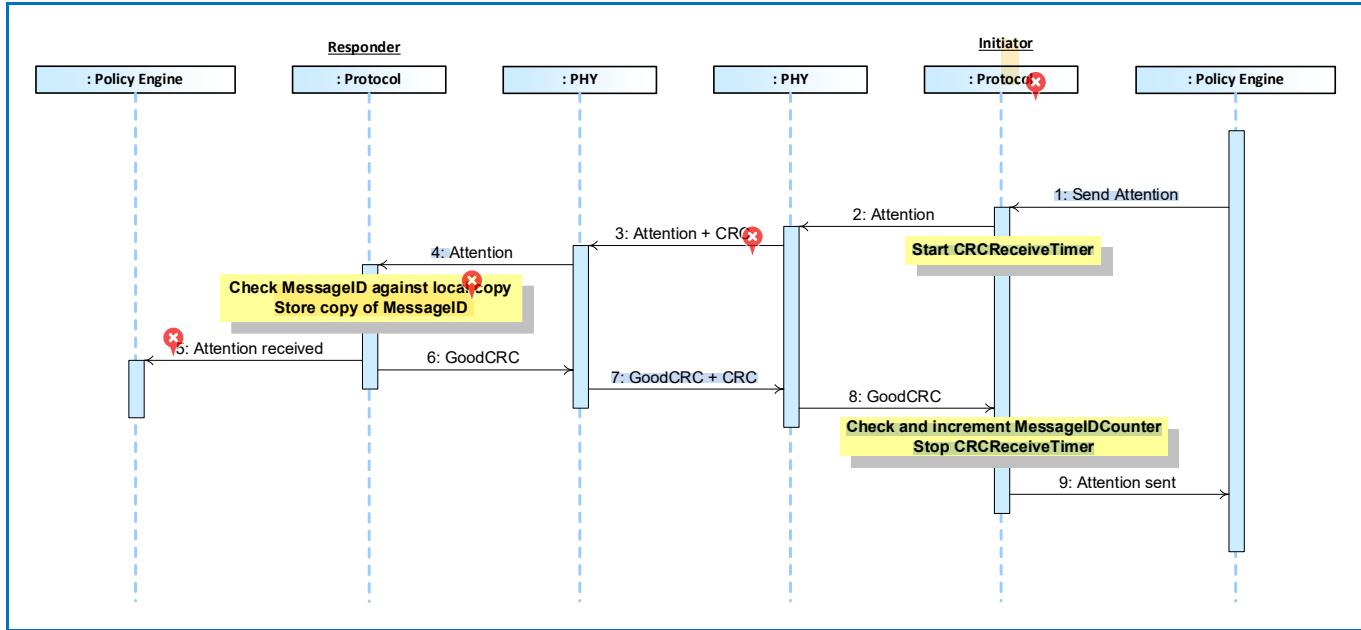


Table 8.145 “Steps for Initiator to Responder Attention” below provides a detailed explanation of what happens at each labeled step in **Figure 8-117 “Initiator to Responder Attention”** above.

Table 8.145 “Steps for Initiator to Responder Attention”

Step	Responder	Initiator
1		The Device Policy Manager requests attention. The Policy Engine tells the Protocol Layer to form an Attention Command request.
2		Protocol Layer creates the Attention Command request and passes to Physical Layer.
3	Physical Layer receives the Attention Command request and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Attention Command request. Starts CRCReceiveTimer .
4	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Attention Command request information to the Policy Engine that consumes it.	
5	The Policy Engine informs the Device Policy Manager	
6	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
7	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.
8		Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.
9		Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Attention Command request was successfully sent.

8.3.2.16 Built in Self-Test (BIST)

8.3.2.16.1 BIST Carrier Mode

The following is an example of a BIST Carrier Mode test between a Tester and a UUT. When the UUT is connected to the Tester the sequence below is executed.

Figure 8-118 “BIST Carrier Mode Test” shows the Messages as they flow across the bus and within the devices. This test enables the measurement of power supply noise and frequency drift.

- 1) Connection is established and stable.
- 2) Tester sends a **BIST** Message with a **BIST Carrier Mode** BIST Data Object.
- 3) UUT answers with a **GoodCRC** Message.
- 4) UUT starts sending the Test Pattern.
- 5) Operator does the measurements.
- 6) The test ends after **tBISTContMode**.

See also [Section 5.9.1 “BIST Carrier Mode”](#) and [Section 6.4.3.1 “BIST Carrier Mode”](#).

Figure 8-118 “BIST Carrier Mode Test”

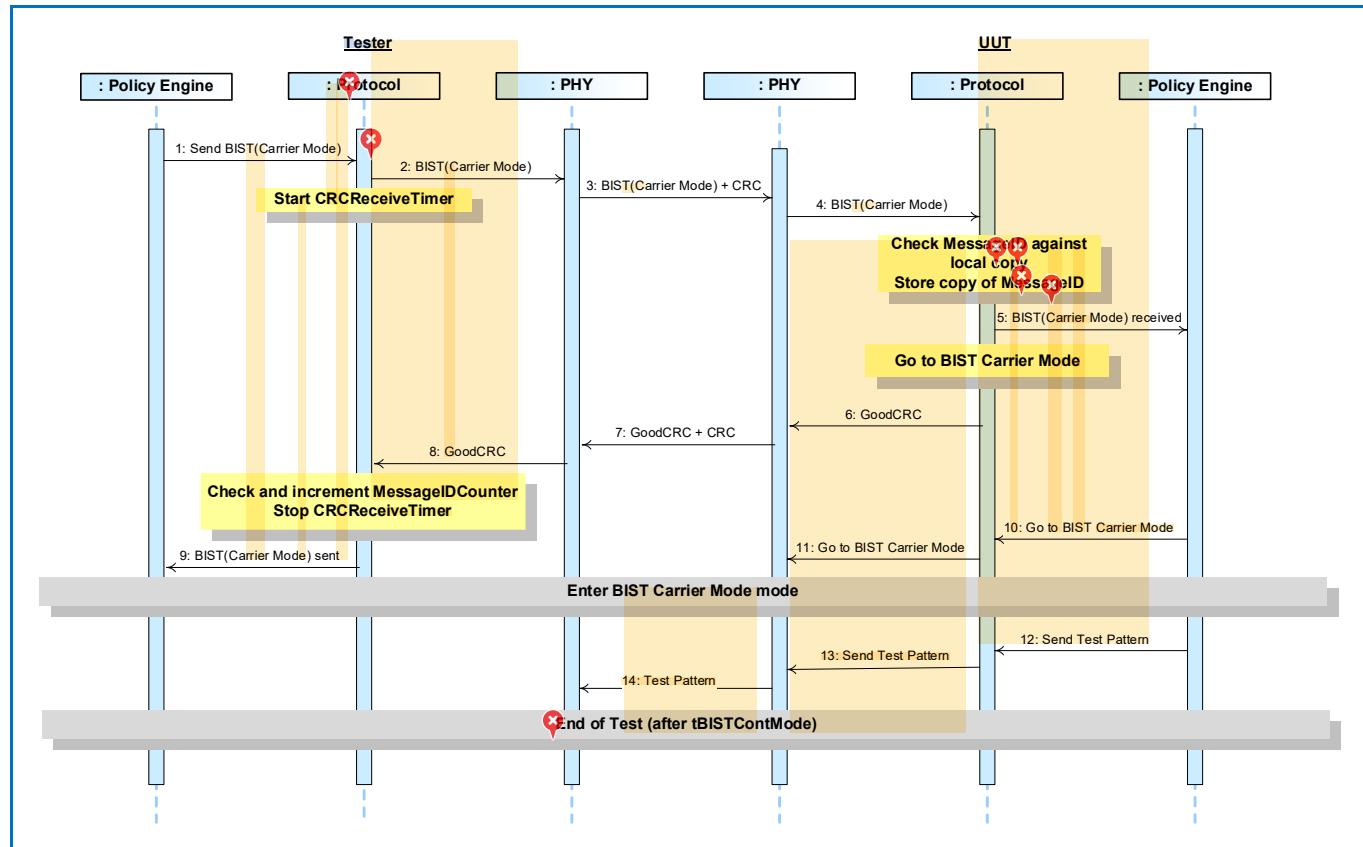


Table 8.146 “Steps for BIST Carrier Mode Test” below provides a detailed explanation of what happens at each labeled step in **Figure 8-118 “BIST Carrier Mode Test”** above.

Table 8.146 “Steps for BIST Carrier Mode Test”

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a BIST Message, with a BIST Data Object of BIST Carrier Mode , to put the UUT into BIST Carrier Mode test mode.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the BIST Message. Starts CRCReceiveTimer .	Physical Layer receives the BIST Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the BIST Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received BIST Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the BIST Message was successfully sent.	
10		Policy Engine tells Protocol Layer to go into BIST Carrier Mode test mode. The Policy Engine goes to BIST Carrier Mode test mode.
11		Protocol Layer tells Physical Layer to go into BIST Carrier Mode test mode.
	UUT enters BIST Carrier Mode test mode	
12		The Policy Engine directs the Protocol Layer to start generation of the Test Pattern.
13		Protocol Layer directs the PHY Layer to generate the Test Pattern.
14	Physical Layer receives the Test Pattern stream.	Physical Layer generates a continuous Test Pattern stream.
The UUT exits BIST Carrier Mode test mode after tBISTContMode.		

8.3.2.16.2 BIST Test Data

✖ The following is an example of a BIST Test Data test between a Tester and a UUT. When the UUT is connected to the Tester the sequence below is executed.

Figure 8-119 “BIST Test Data Test” shows the Messages as they flow across the bus and within the devices.

- 1) Connection is established and stable.
- 2) Tester sends a **BIST** Message with a **BIST Test Data** BIST Data Object.
- 3) UUT answers with a **GoodCRC** Message.
- 4) Steps 2 and 3 are repeated any number of times.
- 5) The test ends after **Hard Reset** Signaling is issued.

See also [Section 5.9.2 “BIST Test Data”](#) and [Section 6.4.3.2 “BIST Test Data”](#).

Figure 8-119 “BIST Test Data Test”

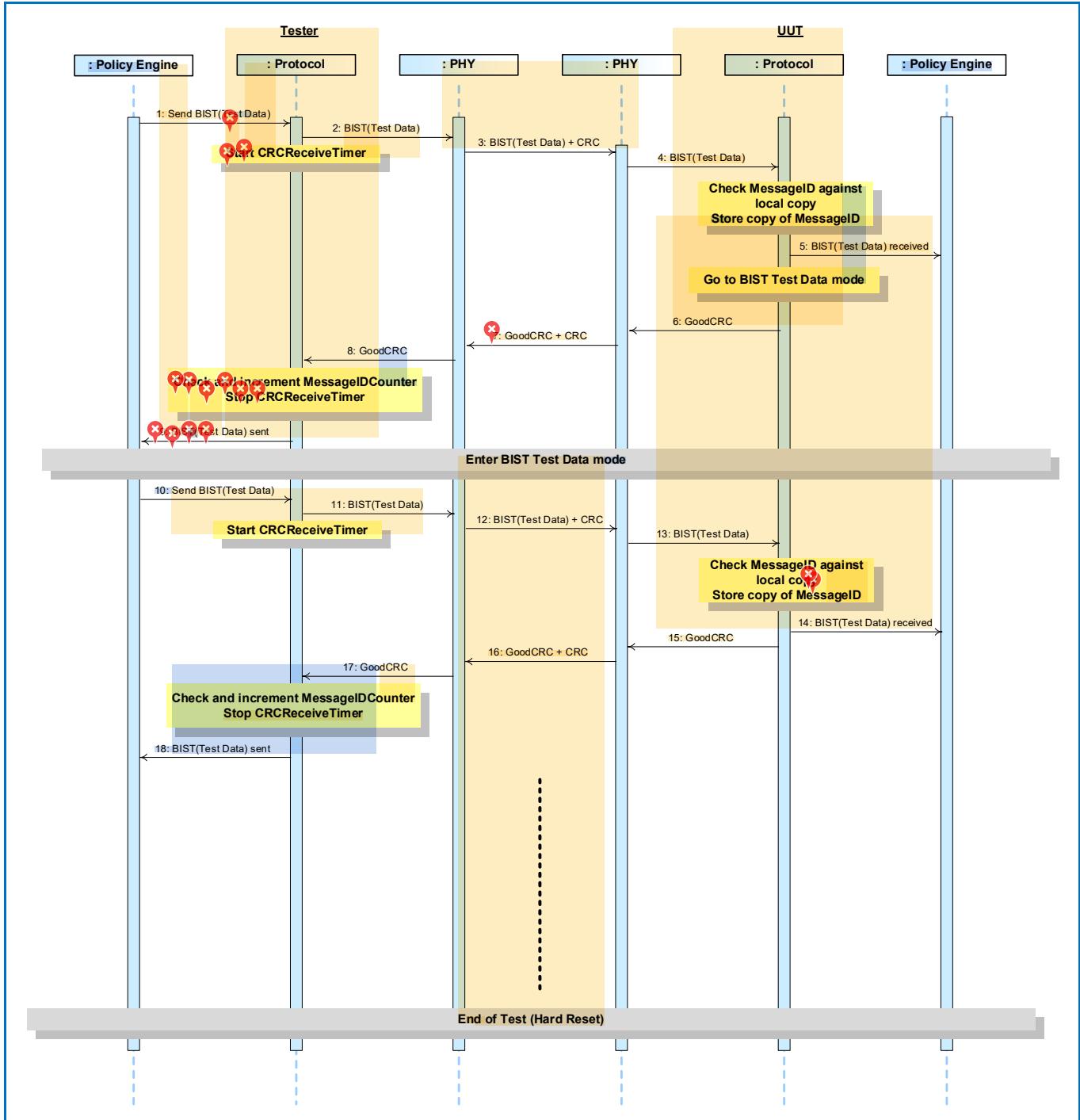


Table 8.147 “Steps for BIST Test Data Test” below provides a detailed explanation of what happens at each labeled step in **Figure 8-119 “BIST Test Data Test”** above.

Table 8.147 “Steps for BIST Test Data Test”

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a BIST Message, with a BIST Data Object of BIST Test Data , to put the UUT into BIST Test Data test mode.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the BIST Message. Starts CRCReceiveTimer .	Physical Layer receives the BIST Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the BIST Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received BIST Message information to the Policy Engine that consumes it. The Policy Engine goes into BIST Test Data Mode where it sends no further Messages except for GoodCRC Messages in response to received Messages (see Section 6.4.3.2 “BIST Test Data”).
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the BIST Message was successfully sent.	
	UUT enters BIST Test Data test mode	
10	The Policy Engine directs the Protocol Layer to generate a BIST Message, with a BIST Data Object of BIST Test Data , to put the UUT into BIST Test Data test mode.	
11	Protocol Layer creates the Message and passes to Physical Layer.	
12	Physical Layer appends CRC and sends the BIST Message. Starts CRCReceiveTimer .	Physical Layer receives the BIST Message and checks the CRC to verify the Message.
13		Physical Layer removes the CRC and forwards the BIST Message to the Protocol Layer.

14		<p>Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value.</p> <p>The Protocol Layer forwards the received <i>BIST</i> Message information to the Policy Engine that consumes it.</p> <p>The Policy Engine goes into BIST Test Data Mode where it sends no further Messages except for <i>GoodCRC</i> Messages in response to received Messages (see Section 6.4.3.2 "BIST Test Data").</p>
15		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
16	Physical Layer receives the <i>GoodCRC</i> and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
17	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
18	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>BIST</i> Message was successfully sent.	
	Repeat steps 10-18 any number of times	
The UUT exits BIST Test Data test mode after a Hard Reset		

8.3.2.16.3 BIST Shared Capacity Test Mode

The following is an example of a BIST Shared Capacity Test Mode test between a Tester and a UUT. When the UUT is connected to the Tester the sequence below is executed.

Figure 8-120 “BIST Share Capacity Mode Test” shows the Messages as they flow across the bus and within the devices. This test places the UUT in a compliance test mode where the maximum source capability is always offered on every port, regardless of the availability of shared power i.e., all shared power management is disabled.

- 1) Connection is established and stable.
- 2) Tester sends a **BIST** Message with a **BIST Shared Test Mode Entry** BIST Data Object.
- 3) UUT answers with a **GoodCRC** Message.
- 4) UUT enters BIST Shared Capacity Test Mode.
- 5) Operator does the measurements.
- 6) Tester sends a **BIST** Message with a **BIST Shared Test Mode Exit** BIST Data Object.
- 7) UUT answers with a **GoodCRC** Message.
- 8) UUT exits BIST Shared Capacity Test Mode.

See also [Section 5.9.1 “BIST Carrier Mode”](#) and [Section 6.4.3.3 “BIST Shared Capacity Test Mode”](#).

Figure 8-120 “BIST Share Capacity Mode Test”

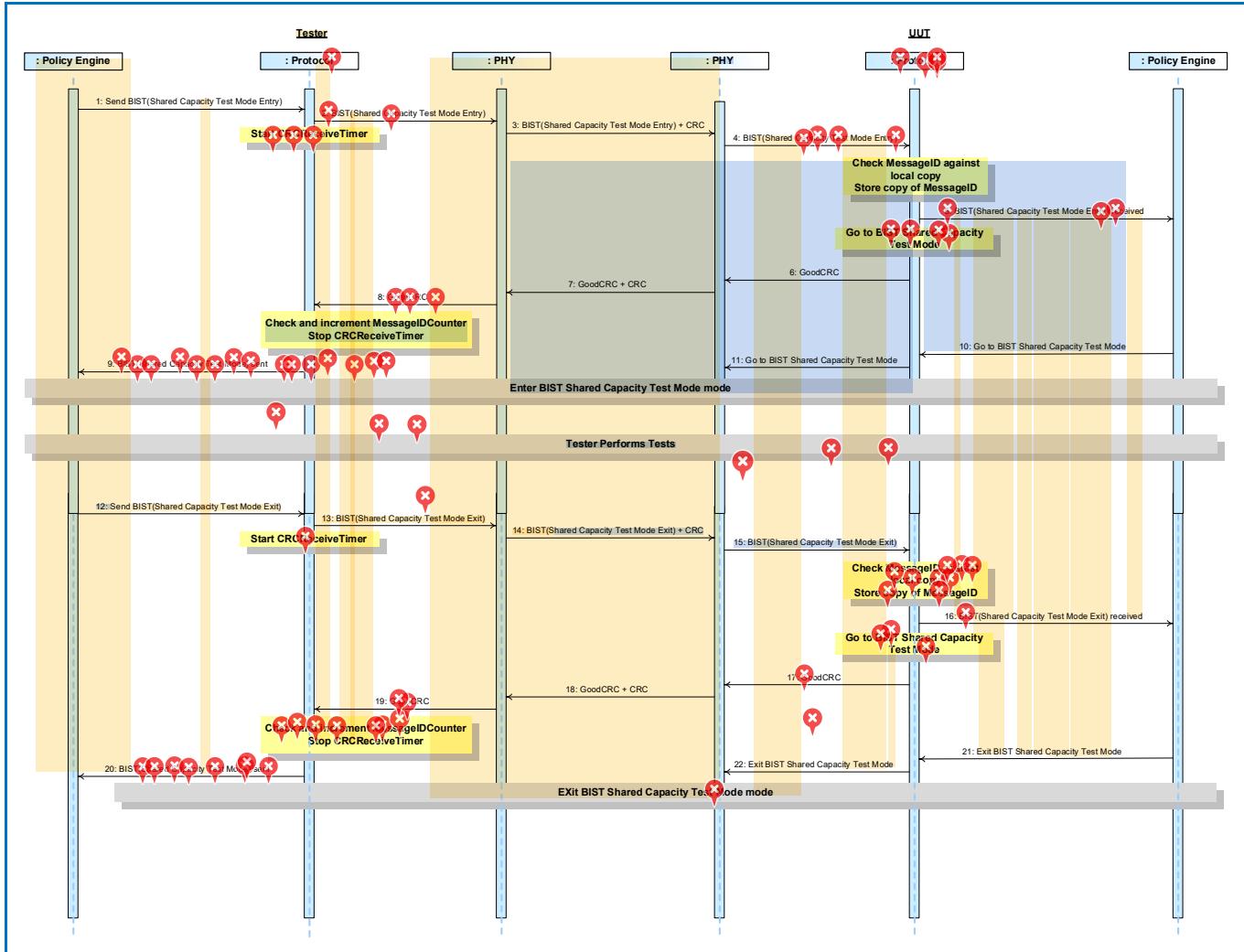


Table 8.148 “Steps for BIST Shared Capacity Test Mode Test” below provides a detailed explanation of what happens at each labeled step in **Figure 8-119 “BIST Test Data Test”** above.

Table 8.148 “Steps for BIST Shared Capacity Test Mode Test”

Step	Tester	UUT
1	The Policy Engine directs the Protocol Layer to generate a BIST Message, with a BIST Data Object of BIST Shared Test Mode Entry , to put the UUT into BIST Shared Capacity Test Mode.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the BIST Message. Starts CRCReceiveTimer .	Physical Layer receives the BIST Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the BIST Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received BIST Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the BIST Message was successfully sent.	
10		Policy Engine tells Protocol Layer to go into BIST Shared Capacity Test Mode. The Policy Engine goes to BIST Shared Capacity Test Mode.
11		Protocol Layer tells Physical Layer to go into BIST Shared Capacity Test Mode.
	UUT enters BIST Shared Capacity Test Mode. Tester performs tests.	
12	The Policy Engine directs the Protocol Layer to generate a BIST Message, with a BIST Data Object of BIST Shared Test Mode Exit , to take the UUT out of BIST Shared Capacity Test Mode.	
13	Protocol Layer creates the Message and passes to Physical Layer.	
14	Physical Layer appends CRC and sends the BIST Message. Starts CRCReceiveTimer .	Physical Layer receives the BIST Message and checks the CRC to verify the Message.
15		Physical Layer removes the CRC and forwards the BIST Message to the Protocol Layer.

16		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>BIST</i> Message information to the Policy Engine that consumes it.
17		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
18	Physical Layer receives the <i>GoodCRC</i> and checks the  CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
19	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
20	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>BIST</i> Message was successfully sent.	
21		Policy Engine tells Protocol Layer to exit BIST Shared Capacity Test Mode. The Policy Engine exits to BIST Shared Capacity Test Mode.
22		Protocol Layer tells Physical Layer to exit BIST Shared Capacity Test Mode.
UUT exits BIST Shared Capacity Test Mode.		

8.3.2.17 Enter USB

8.3.2.17.1 UFP Entering USB4® Mode

8.3.2.17.1.1 UFP Entering USB4® Mode (Accept)

This is an example of an Enter USB operation where the DFP requests [USB4] mode when this is a **Valid** mode of operation for the UFP. [Figure 8-121 “UFP Entering USB4® Mode \(Accept\)”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Enter USB process.

Figure 8-121 “UFP Entering USB4® Mode (Accept)”

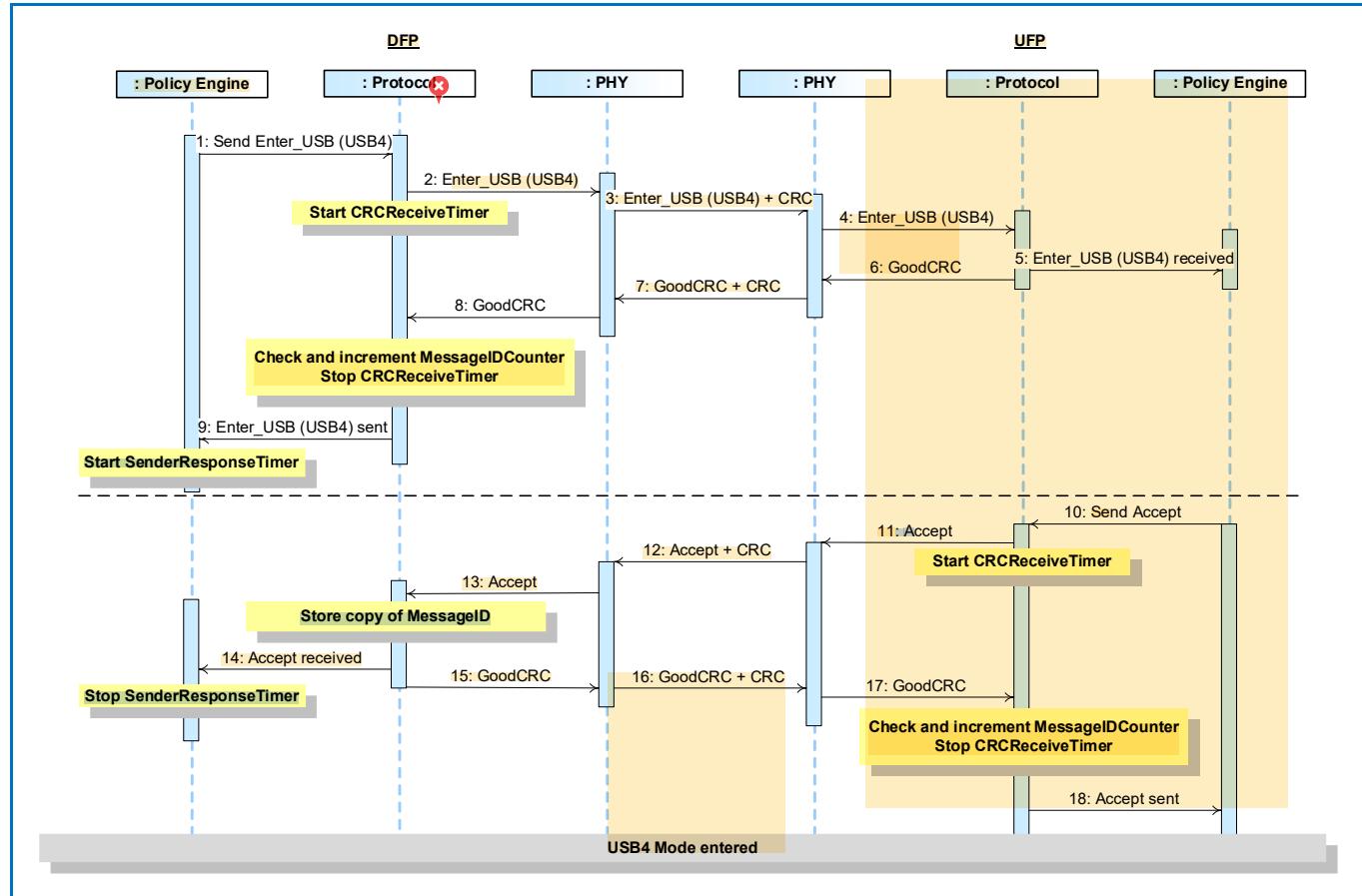


Table 8.149 "Steps for UFP USB4® Mode Entry (Accept)" below provides a detailed explanation of what happens at each labeled step in **Figure 8-121 "UFP Entering USB4® Mode (Accept)"** above.

Table 8.149 "Steps for UFP USB4® Mode Entry (Accept)"

Step	DFP	UFP
1	The Policy Engine directs the Protocol Layer to generate an <i>Enter_USB</i> Message to request entry to [USB4] mode.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the <i>Enter_USB</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the <i>Enter_USB</i> Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Enter_USB</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Enter_USB</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> and checks the <i>CRC</i> to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Enter_USB</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine tells the Protocol Layer to form an <i>Accept</i> Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Message. Starts <i>CRCReceiveTimer</i> .
13	Protocol Layer stores the <i>MessageID</i> of the incoming Message.	
14	The Protocol Layer forwards the received <i>Accept</i> Message information to the Policy Engine that consumes it.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.

17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		<p>Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i>. Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent.</p>
Both Port Partners enter [USB4] operation.		

8.3.2.17.1.2

UFP Entering USB4® Mode (Reject)

This is an example of an Enter USB operation where the DFP requests [USB4] mode when this is an **Invalid** mode of operation for the UFP. [Figure 8-122 “UFP Entering USB4® Mode \(Reject\)”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Enter USB process.

[Figure 8-122 “UFP Entering USB4® Mode \(Reject\)”](#)

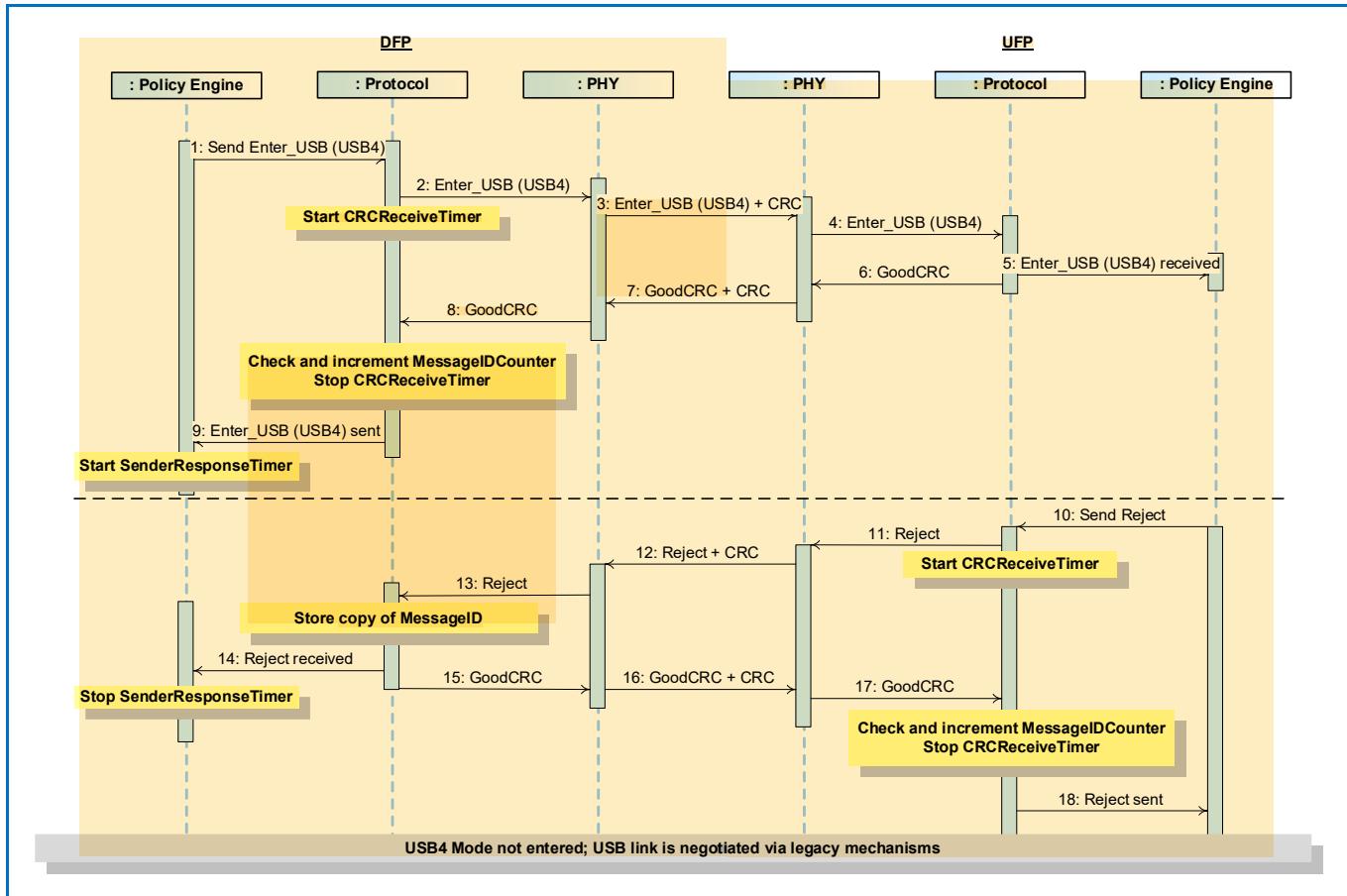


Table 8.150 “Steps for UFP USB4® Mode Entry (Reject)” below provides a detailed explanation of what happens at each labeled step in **Figure 8-122 “UFP Entering USB4® Mode (Reject)”** above.

Table 8.150 “Steps for UFP USB4® Mode Entry (Reject)”

Step	DFP	UFP
1	The Policy Engine directs the Protocol Layer to generate an <i>Enter_USB</i> Message to request entry to [USB4] mode.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the <i>Enter_USB</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the <i>Enter_USB</i> Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Enter_USB</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Enter_USB</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> and checks the <i>CRC</i> to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Enter_USB</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine tells the Protocol Layer to form an <i>Reject</i> Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Message. Starts <i>CRCReceiveTimer</i> .
13	Protocol Layer stores the <i>MessageID</i> of the incoming Message.	
14	The Protocol Layer forwards the received <i>Reject</i> Message information to the Policy Engine that consumes it.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.

17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		<p>Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i>. Protocol Layer informs the Policy Engine that the <i>Reject</i> Message was successfully sent.</p>
Port Partners do not enter [USB4] operation.		

8.3.2.17.1.3

UFP Entering USB4® Mode (Wait)

This is an example of an Enter USB operation where the DFP requests [USB4] mode when this is not possible for the UFP at this time. [Figure 8-123 “UFP Entering USB4® Mode \(Wait\)”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Enter USB process.

Figure 8-123 “UFP Entering USB4® Mode (Wait)”

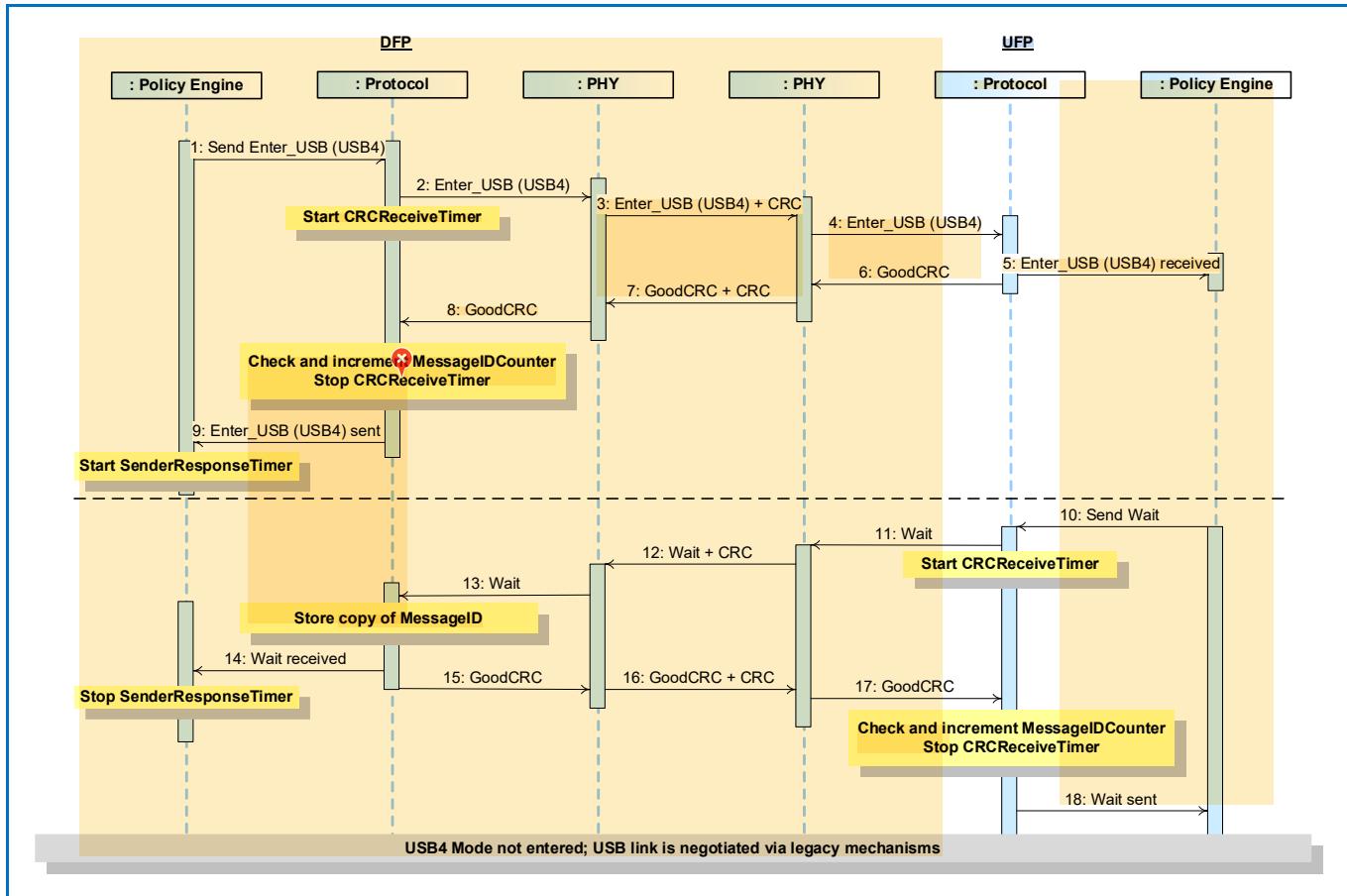


Table 8.151 "Steps for UFP USB4® Mode Entry (Wait)" below provides a detailed explanation of what happens at each labeled step in **Figure 8-123 "UFP Entering USB4® Mode (Wait)"** above.

Table 8.151 "Steps for UFP USB4® Mode Entry (Wait)"

Step	DFP	UFP
1	The Policy Engine directs the Protocol Layer to generate an Enter_USB Message to request entry to [USB4] mode.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Enter_USB Message. Starts CRCReceiveTimer .	Physical Layer receives the Enter_USB Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the Enter_USB Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Enter_USB Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Enter_USB Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine tells the Protocol Layer to form an Wait Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Message. Starts CRCReceiveTimer .
13	Protocol Layer stores the MessageID of the incoming Message.	
14	The Protocol Layer forwards the received Wait Message information to the Policy Engine that consumes it.	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.

17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		<p>Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i>. ✖ Protocol Layer informs the Policy Engine that the <i>Wait</i> Message was successfully sent.</p>
Port Partners do not enter [USB4] operation.		

8.3.2.17.2 Cable Plug Entering USB4® Mode

8.3.2.17.2.1 Cable Plug Entering USB4® Mode (Accept)

This is an example of an Enter USB operation where the DFP requests [USB4] mode when this is a **Valid** mode of operation for the Cable Plug. [Figure 8-124 “Cable Plug Entering USB4® Mode \(Accept\)](#) shows the Messages as they flow across the bus and within the devices to accomplish the Enter USB process.

Figure 8-124 “Cable Plug Entering USB4® Mode (Accept)”

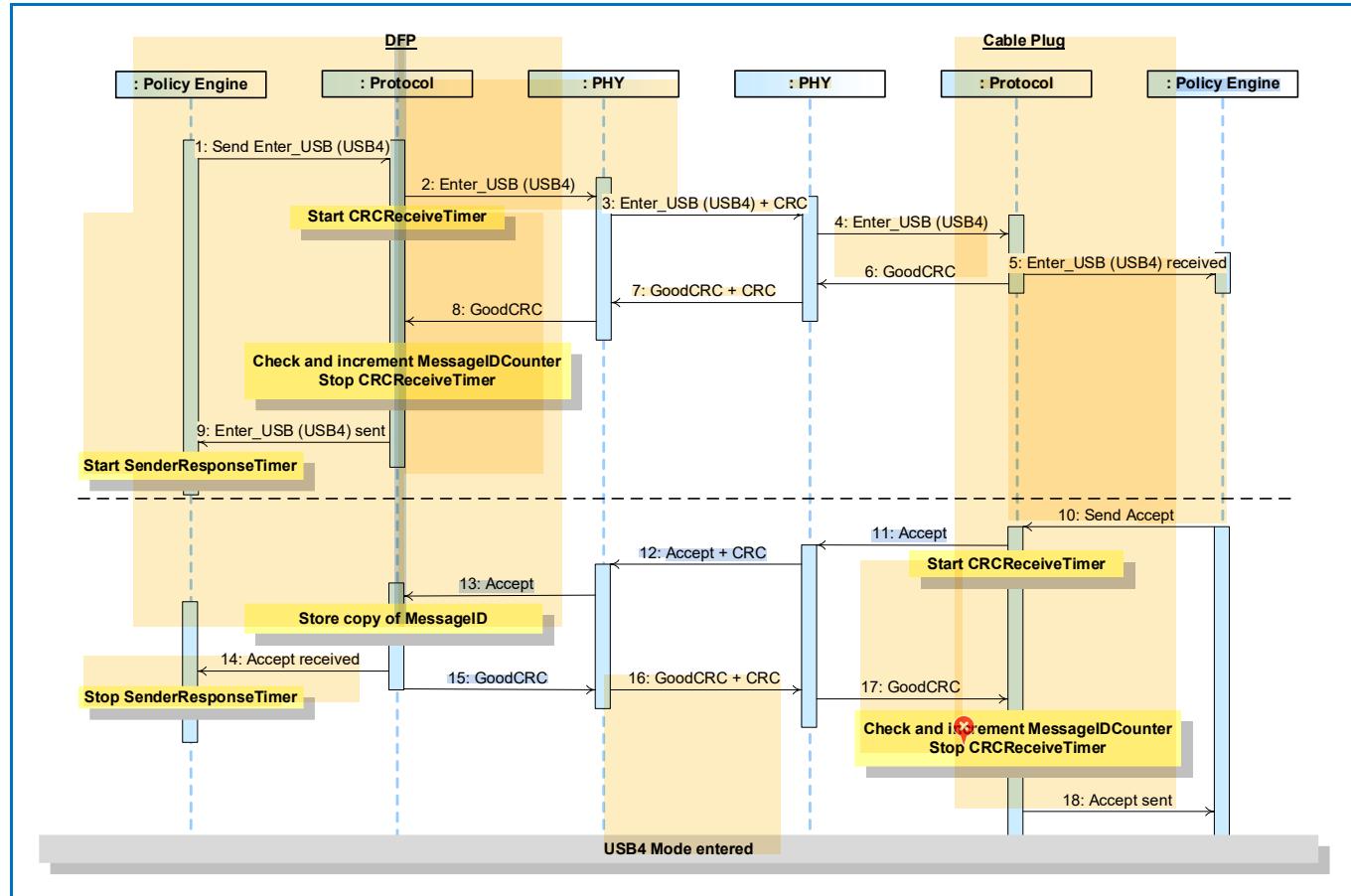


Table 8.152 "Steps for Cable Plug USB4® Mode Entry (Accept)" below provides a detailed explanation of what happens at each labeled step in **Figure 8-124 "Cable Plug Entering USB4® Mode (Accept)"** above.

Table 8.152 "Steps for Cable Plug USB4® Mode Entry (Accept)"

Step	DFP	Cable Plug
1	The Policy Engine directs the Protocol Layer to generate an Enter_USB Message to request entry to [USB4] mode.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Enter_USB Message. Starts CRCReceiveTimer .	Physical Layer receives the Enter_USB Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the Enter_USB Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Enter_USB Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Enter_USB Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine tells the Protocol Layer to form an Accept Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Message. Starts CRCReceiveTimer .
13	Protocol Layer stores the MessageID of the incoming Message.	
14	The Protocol Layer forwards the received Accept Message information to the Policy Engine that consumes it.	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.

17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		<p>Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i>. Protocol Layer informs the Policy Engine that the <i>Accept</i> Message was successfully sent.</p>
Cable Plug enters [USB4] operation.		

8.3.2.17.2.2

Cable Plug Entering USB4® Mode (Reject)

This is an example of an Enter USB operation where the DFP requests [USB4] mode when this is an **Invalid** mode of operation for the Cable Plug. [Figure 8-125 “Cable Plug Entering USB4® Mode \(Reject\)”](#) shows the Messages as they flow across the bus and within the devices to accomplish the Enter USB process.

[Figure 8-125 “Cable Plug Entering USB4® Mode \(Reject\)”](#)

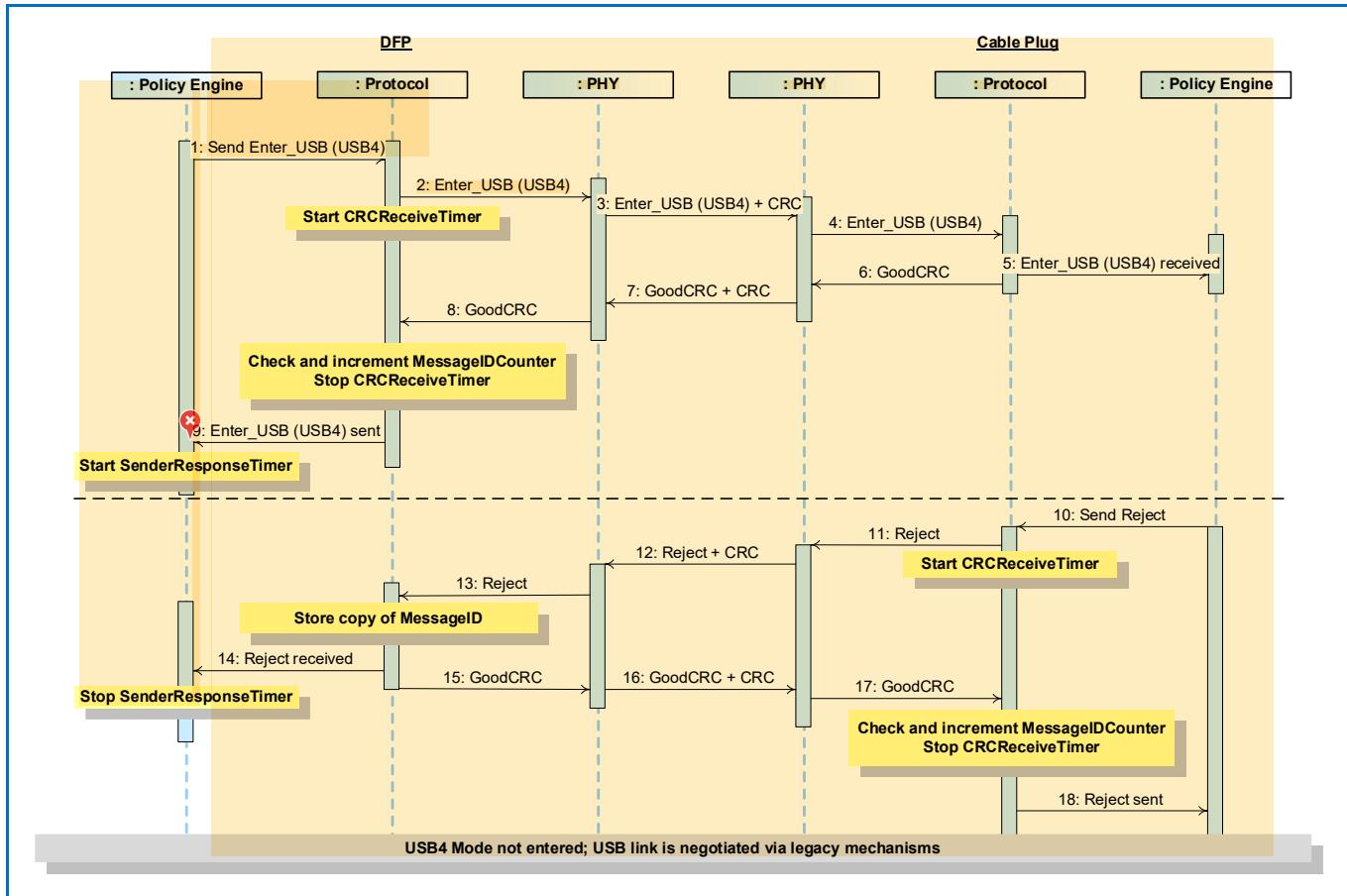


Table 8.153 “Steps for Cable Plug USB4® Mode Entry (Reject)” below provides a detailed explanation of what happens at each labeled step in **Figure 8-125 “Cable Plug Entering USB4® Mode (Reject)”** above.

Table 8.153 “Steps for Cable Plug USB4® Mode Entry (Reject)”

Step	DFP	Cable Plug
1	The Policy Engine directs the Protocol Layer to generate an <i>Enter_USB</i> Message to request entry to [USB4] mode.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the <i>Enter_USB</i> Message. Starts <i>CRCReceiveTimer</i> .	Physical Layer receives the <i>Enter_USB</i> Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the <i>Enter_USB</i> Message to the Protocol Layer.
5		Protocol Layer checks the <i>MessageID</i> in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received <i>Enter_USB</i> Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.
7	Physical Layer receives the <i>GoodCRC</i> and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the <i>GoodCRC</i> Message.
8	Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Enter_USB</i> Message was successfully sent. Policy Engine starts <i>SenderResponseTimer</i> .	
10		Policy Engine tells the Protocol Layer to form an <i>Reject</i> Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Message. Starts <i>CRCReceiveTimer</i> .
13	Protocol Layer stores the <i>MessageID</i> of the incoming Message.	
14	The Protocol Layer forwards the received <i>Reject</i> Message information to the Policy Engine that consumes it.	
15	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.

17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		<p>Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i>. Protocol Layer informs the Policy Engine that the <i>Reject</i> Message was successfully sent.</p>
Cable Plug does not enter [USB4] operation.		

8.3.2.17.2.3

Cable Plug Entering USB4® Mode (Wait)

This is an example of an Enter USB operation where the DFP requests [USB4] mode when this is not possible for the Cable Plug at this time. [Figure 8-126 “Cable Plug Entering USB4® Mode \(Wait\)](#) shows the Messages as they flow across the bus and within the devices to accomplish the Enter USB process.

[Figure 8-126 “Cable Plug Entering USB4® Mode \(Wait\)”](#)

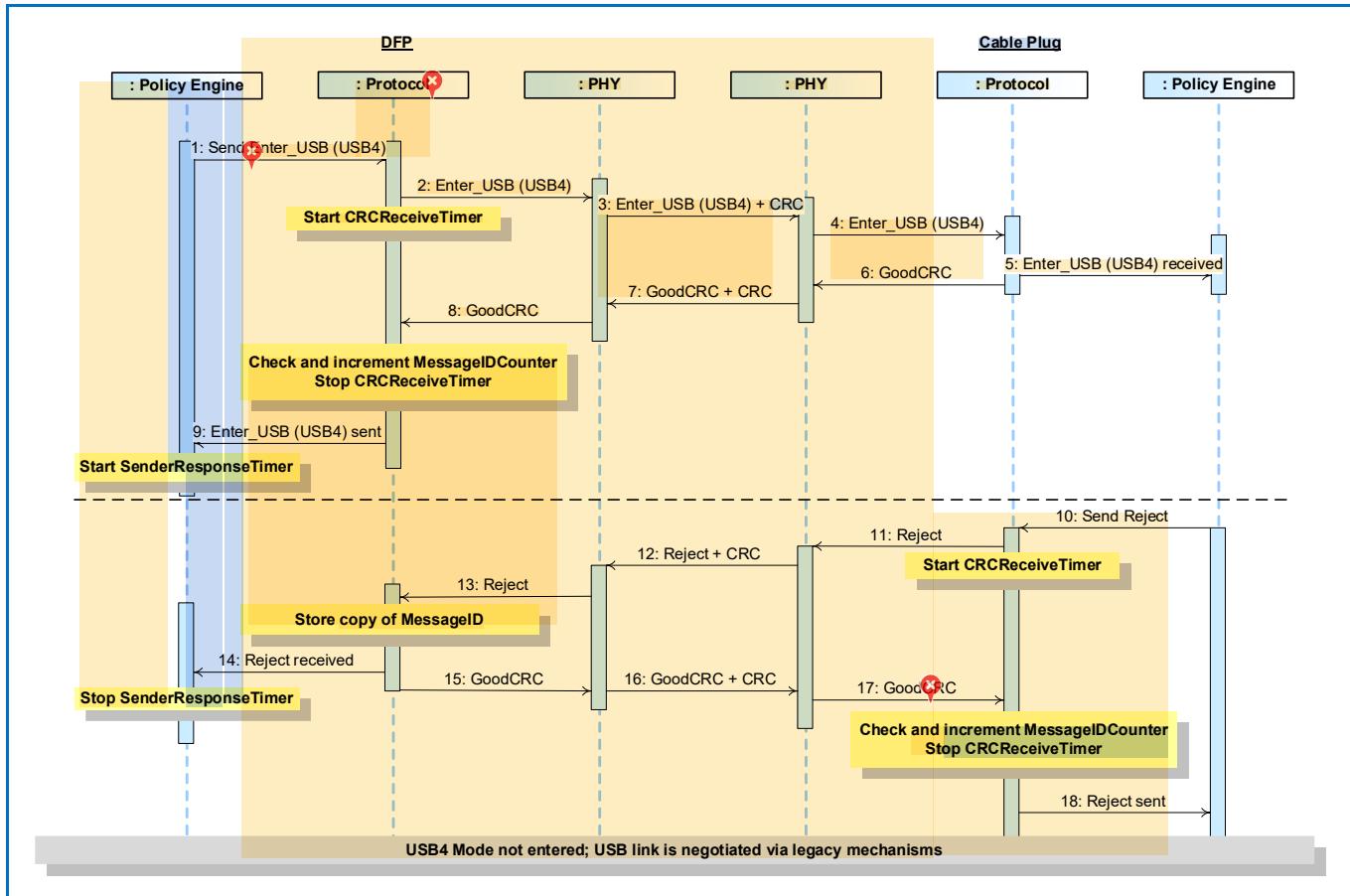


Table 8.154 “Steps for Cable Plug USB4® Mode Entry (Wait)” below provides a detailed explanation of what happens at each labeled step in **Figure 8-126 “Cable Plug Entering USB4® Mode (Wait)”** above.

Table 8.154 “Steps for Cable Plug USB4® Mode Entry (Wait)”

Step	DFP	Cable Plug
1	The Policy Engine directs the Protocol Layer to generate an Enter_USB Message to request entry to [USB4] mode.	
2	Protocol Layer creates the Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Enter_USB Message. Starts CRCReceiveTimer .	Physical Layer receives the Enter_USB Message and compares the CRC it calculated with the one sent to verify the Message.
4		Physical Layer removes the CRC and forwards the Enter_USB Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the received Enter_USB Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Enter_USB Message was successfully sent. Policy Engine starts SenderResponseTimer .	
10		Policy Engine tells the Protocol Layer to form an Wait Message.
11		Protocol Layer creates the Message and passes to Physical Layer.
12	Physical Layer receives the Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Message. Starts CRCReceiveTimer .
13	Protocol Layer stores the MessageID of the incoming Message.	
14	The Protocol Layer forwards the received Wait Message information to the Policy Engine that consumes it.	
15	Protocol Layer generates a GoodCRC Message and passes it Physical Layer.	
16	Physical Layer appends a CRC and sends the GoodCRC Message.	Physical Layer receives GoodCRC Message and compares the CRC it calculated with the one sent to verify the Message.

17		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
18		<p>Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i>. Protocol Layer informs the Policy Engine that the <i>Wait</i> Message was successfully sent.</p>
Cable Plug does not enter [USB4] operation.		

8.3.2.18 Unstructured Vendor Defined Messages

8.3.2.18.1 Unstructured VDM

Figure 8-127 “Unstructured VDM Message Sequence” shows an example sequence of an Unstructured VDM Transaction between a DFP and UFP. The figure below shows the messages as they flow across the bus after UFP Enters into modal operation.

Figure 8-127 “Unstructured VDM Message Sequence”

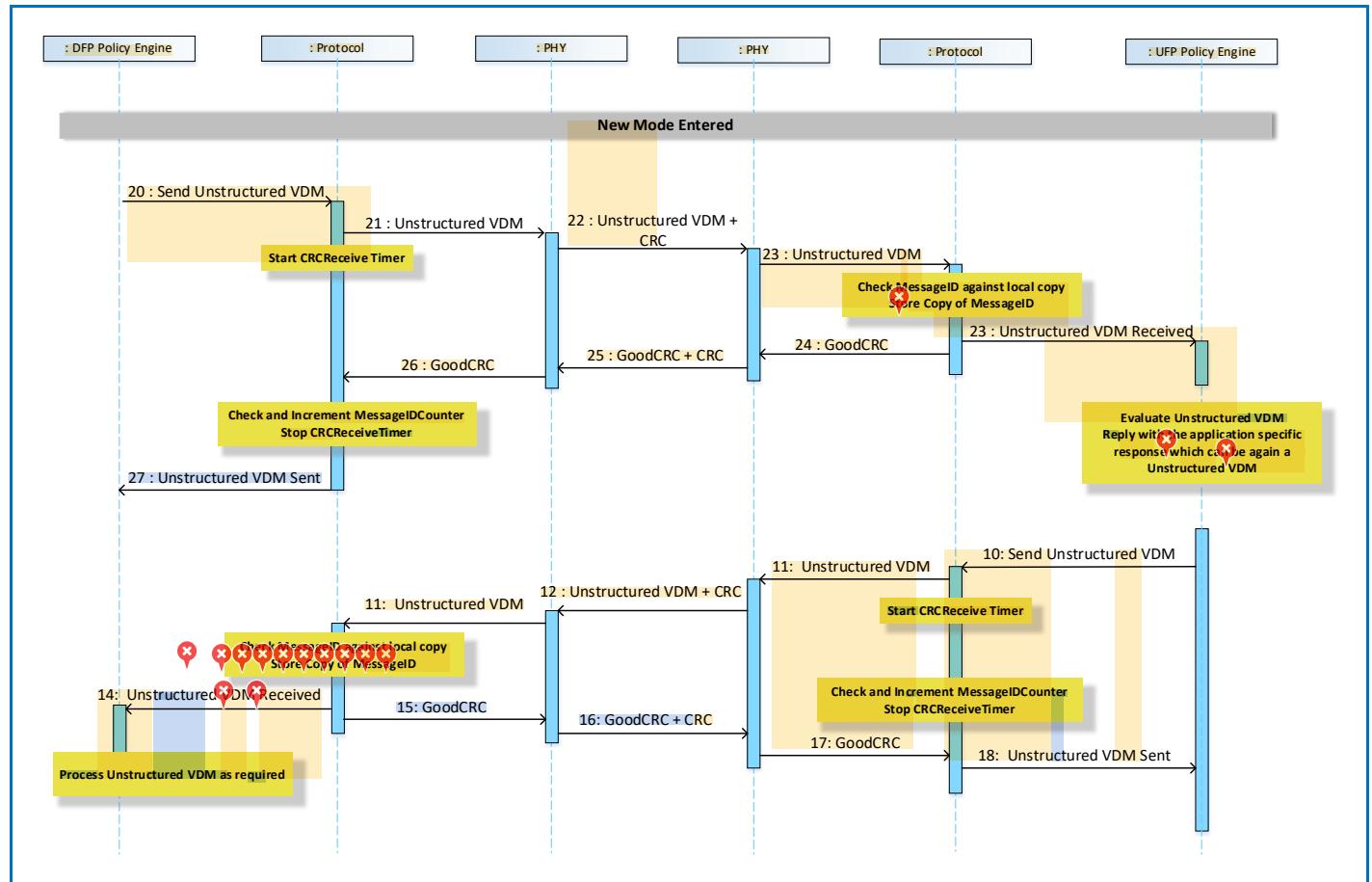


Table 8.155 “Steps for Unstructured VDM Message Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-127 “Unstructured VDM Message Sequence”** above.

Table 8.155 “Steps for Unstructured VDM Message Sequence”

Step	DFP	UFP
1	The DFP has an Explicit Contract and has entered an Active Mode with the UFP. The Policy Engine directs the Protocol Layer to send an Unstructured Vendor Defined Message.	The UFP has an Explicit Contract and has entered an Active Mode with the UFP
2	Protocol Layer creates the Unstructured Vendor Defined Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Unstructured Vendor Defined Message. Starts CRCReceiveTimer .	Physical Layer receives the Unstructured Vendor Defined Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Unstructured Vendor Defined Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the Unstructured Vendor Defined Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Unstructured Vendor Defined Message was successfully sent.	
10		In this example the Vendor protocol requires a response. The Policy Engine tells the Protocol Layer to form an Unstructured Vendor Defined Message.
11		Protocol Layer creates the Unstructured Vendor Defined Message and passes to Physical Layer.
12	Physical Layer receives the Unstructured Vendor Defined Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Unstructured Vendor Defined Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the Unstructured Vendor Defined Message information to the Policy Engine that consumes it.	

14	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
15	Physical Layer appends a CRC and sends the <i>GoodCRC?</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
16		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
17		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the Unstructured <i>Vendor_Defined</i> Message was successfully sent.

8.3.2.18.2 VDEM

Figure 8-128 “VDEM Message Sequence” shows an example sequence of an VDEM Transaction between a DFP and UFP. The figure below shows the messages as they flow across the bus after UFP Enters into modal operation.

Figure 8-128 “VDEM Message Sequence”

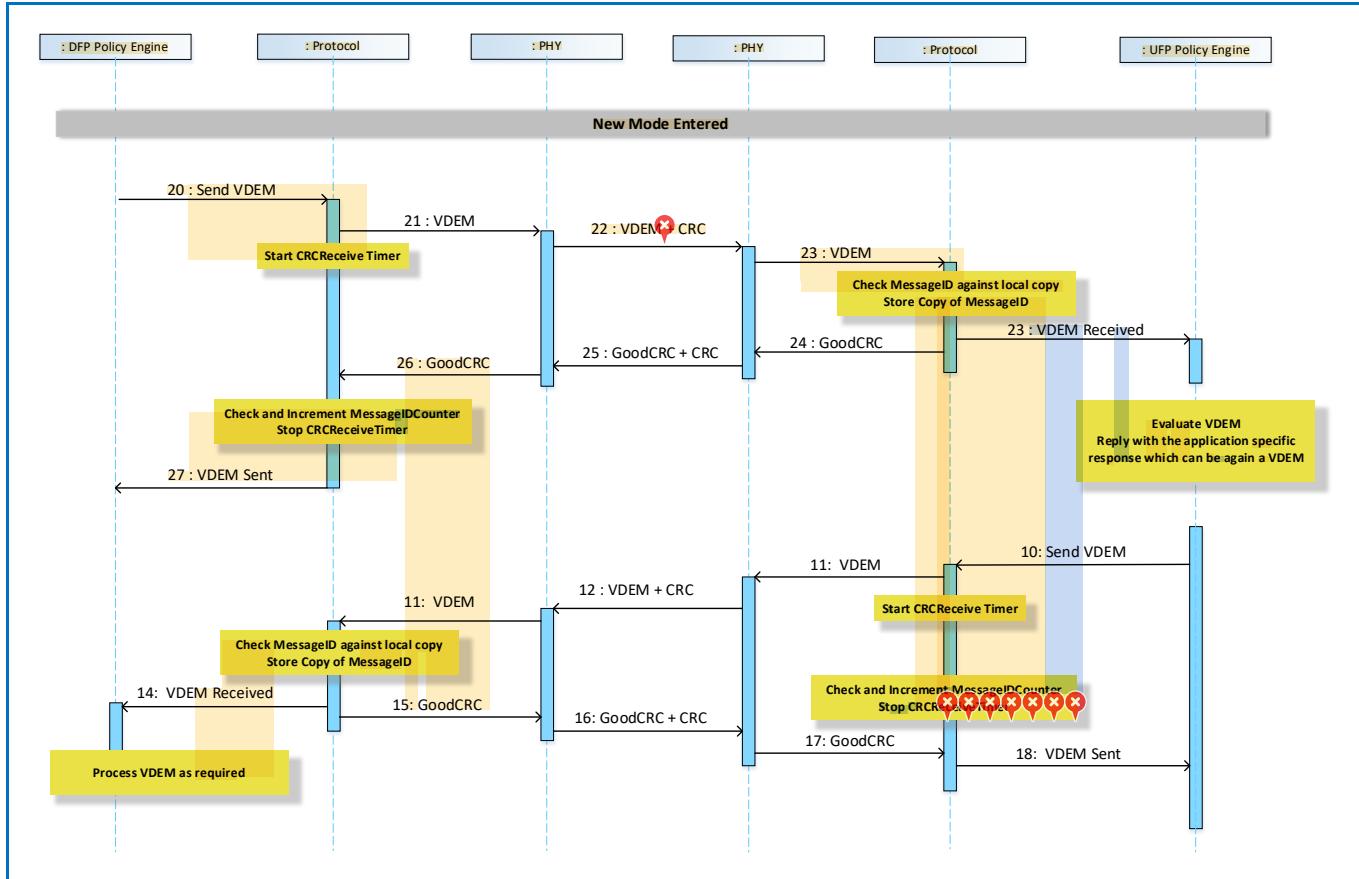


Table 8.156 “Steps for VDEM Message Sequence” below provides a detailed explanation of what happens at each labeled step in **Figure 8-128 “VDEM Message Sequence”** above.

Table 8.156 “Steps for VDEM Message Sequence”

Step	DFP	UFP
1	The DFP has an Explicit Contract and has entered an <i>Active Mode</i> with the UFP. The Policy Engine directs the Protocol Layer to send a Vendor Defined Extended Message.	The UFP has an Explicit Contract and has entered an <i>Active Mode</i> with the UFP
2	Protocol Layer creates the Vendor Defined Extended Message and passes to Physical Layer.	
3	Physical Layer appends CRC and sends the Vendor Defined Extended Message. Starts CRCReceiveTimer .	Physical Layer receives the Vendor Defined Extended Message and checks the CRC to verify the Message.
4		Physical Layer removes the CRC and forwards the Vendor Defined Extended Message to the Protocol Layer.
5		Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the Vendor Defined Extended Message information to the Policy Engine that consumes it.
6		Protocol Layer generates a GoodCRC Message and passes it Physical Layer.
7	Physical Layer receives the GoodCRC Message and checks the CRC to verify the Message.	Physical Layer appends CRC and sends the GoodCRC Message.
8	Physical Layer removes the CRC and forwards the GoodCRC Message to the Protocol Layer.	
9	Protocol Layer verifies and increments the MessageIDCounter and stops CRCReceiveTimer . Protocol Layer informs the Policy Engine that the Vendor Defined Extended Message was successfully sent.	
10		In this example the Vendor protocol requires a response. The Policy Engine tells the Protocol Layer to form a Vendor Defined Extended Message.
11		Protocol Layer creates the Vendor Defined Extended Message and passes to Physical Layer.
12	Physical Layer receives the Vendor Defined Extended Message and compares the CRC it calculated with the one sent to verify the Message.	Physical Layer appends a CRC and sends the Vendor Defined Extended Message. Starts CRCReceiveTimer .
13	Protocol Layer checks the MessageID in the incoming Message is different from the previously stored value and then stores a copy of the new value. The Protocol Layer forwards the Vendor Defined Extended Message information to the Policy Engine that consumes it.	

14	Protocol Layer generates a <i>GoodCRC</i> Message and passes it Physical Layer.	
15	Physical Layer appends a CRC and sends the <i>GoodCRC</i> Message.	Physical Layer receives <i>GoodCRC</i> Message and compares the CRC it calculated with the one sent to verify the Message.
16		Physical Layer removes the CRC and forwards the <i>GoodCRC</i> Message to the Protocol Layer.
17		Protocol Layer verifies and increments the <i>MessageIDCounter</i> and stops <i>CRCReceiveTimer</i> . Protocol Layer informs the Policy Engine that the <i>Vendor_Defined_Extended</i> Message was successfully sent.

8.3.3 State Diagrams

8.3.3.1 Introduction to state diagrams used in Chapter 8

The state diagrams defined in [Section 8.3.3 “State Diagrams”](#) are **Normative** and **Shall** define the operation of the Power Delivery Policy Engine. Note that these state diagrams are not intended to replace a well written and robust design.

Figure 8-129 “Outline of States”

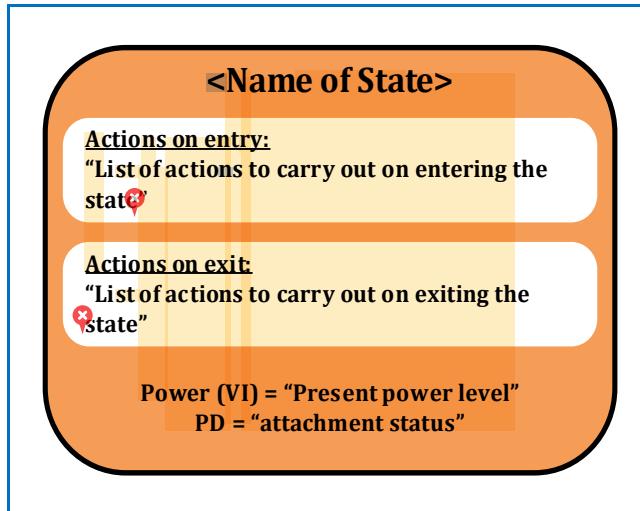


Figure 8-129 “Outline of States” shows an outline of the states defined in the following sections. At the top there is the name of the state. This is followed by “Actions on entry” a list of actions carried out on entering the state. If there are also “Actions on exit” a list of actions carried out on exiting the state, then these are listed as well; otherwise, this box is omitted from the state. At the bottom the status of PD is listed:

- “Power” which indicates the present output power for a Source Port or input power for a Sink Port.
- “PD” which indicates the present Attachment status either “Attached”, “Detached”, or “unknown”.

Transitions from one state to another are indicated by arrows with the conditions listed on the arrow. Where there are multiple conditions, these are connected using either a logical OR “|” or a logical AND “&”.

In some cases, there are transitions which can occur from any state to a particular state. These are indicated by an arrow which is unconnected to a state at one end, but with the other end (the point) connected to the final state.

In some state diagrams it is necessary to enter or exit from states in other diagrams (e.g., Source Port or Sink Port state diagrams). [Figure 8-130 “References to states”](#) indicates how such references are made. The reference is indicated with a hatched box. The box contains the name of the state and whether the state is a DFP or UFP. It has also been necessary to indicate conditional entry to either Source Port or Sink Port state diagrams. This is achieved by the use of a bulleted list indicating the pre-conditions (see example in [Figure 8-131 “Example of state reference with conditions”](#)). It is also possible that the entry and return states are the same. [Figure 8-132 “Example of state reference with the same entry and exit”](#) indicates a state reference where each referenced state corresponds to either the entry state or the exit state.

Figure 8-130 “References to states”

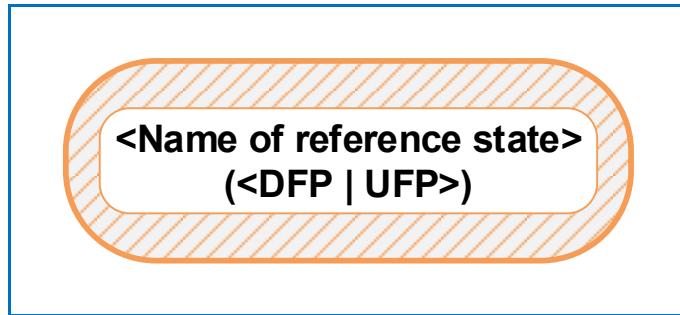


Figure 8-131 “Example of state reference with conditions”

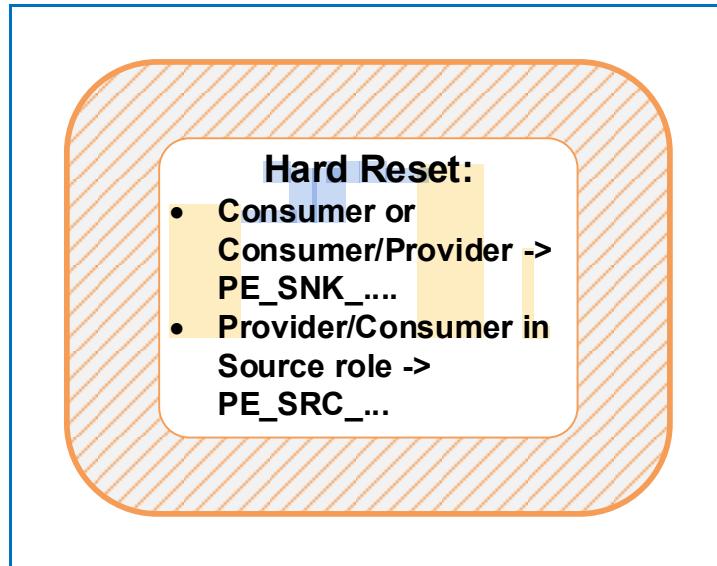
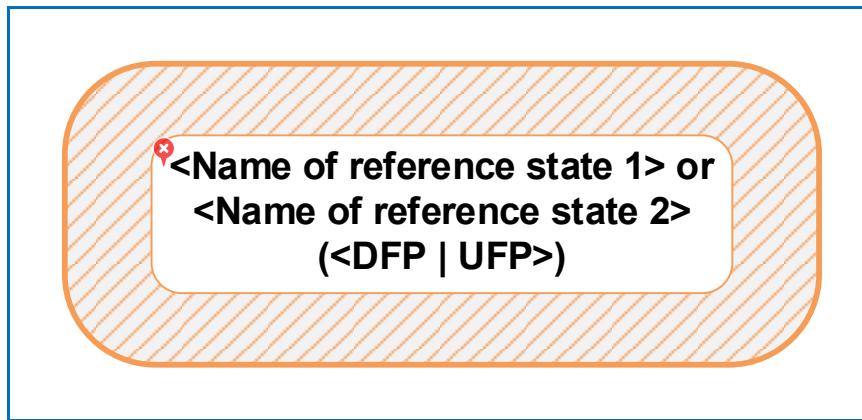


Figure 8-132 “Example of state reference with the same entry and exit”



Timers are included in many of the states. Timers are initialized (set to their starting condition) and run (timer is counting) in the particular state it is referenced. As soon as the state is exited then the timer is no longer active. Where the timers continue to run outside of the state (such as the *NoResponseTimer*), this is called out in the text. Timeouts of the timers are listed as conditions on state transitions.

The **SenderResponseTimer** is a special case, as it **May** be stopped and started from outside the states in which it is used. To allow this to be done without over-complicating the state diagrams, the **SenderResponseTimer** is described with its own state diagram ([Figure 8-133 “SenderResponseTimer Policy Engine State Diagram”](#)). The control of this Timer is shared between the Policy Engine and the Chunking Layer.

Conditions listed on state transitions will come from one of three sources and, when there is a conflict, **Should** be serviced in the following order:

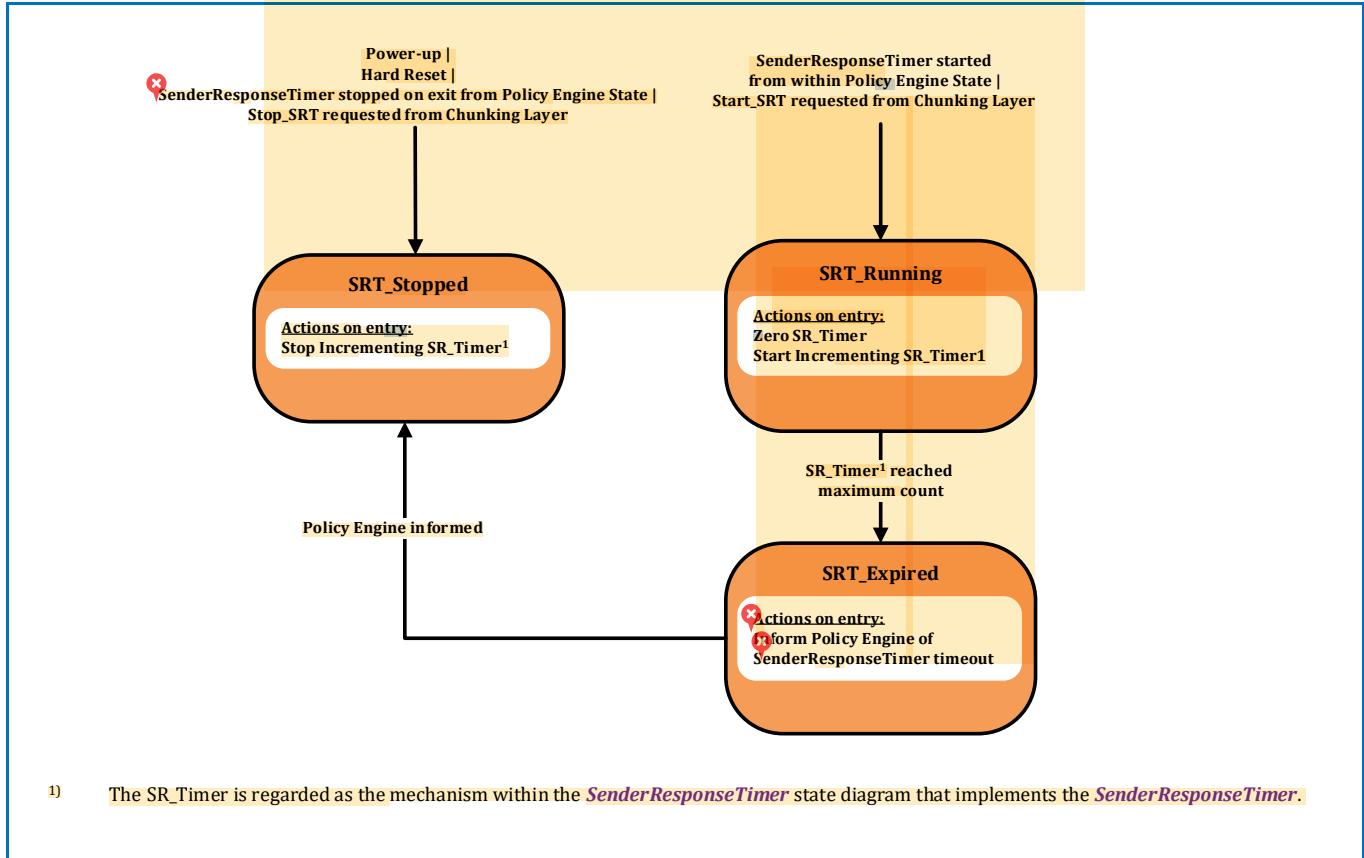
- 1) Message and related indications passed up to the Policy Engine from the Protocol Layer (Message sent; Message received etc.).
- 2) Events triggered within the Policy Engine e.g., timer timeouts.
- 3) Information and requests coming from the Device Policy manager relating either to Local Policy, or to other modules which the Device Policy Manager controls such as power supply and USB-C Port Control.

Note: The following state diagrams are not intended to cover all possible corner cases that could be encountered. For example, where an outgoing Message is **Discarded**, due to an incoming Message by the Protocol Layer (see [Section 6.12.2.3 “Protocol Layer Message Reception”](#)) it will be necessary for the higher layers of the system to handle a retry of the Message sequence that was being initiated, after first handling the incoming Message.

8.3.3.1.1 SenderResponseTimer State Diagram

Figure 8-133 "SenderResponseTimer Policy Engine State Diagram" below shows the state diagram for the Policy Engine in a Source or a Sink Port. The following sections describe operation in each of the states.

Figure 8-133 "SenderResponseTimer Policy Engine State Diagram"



8.3.3.1.1.1 SRT_Stopped State

The **SRT_Stopped** State **Shall** be the starting state for the *SenderResponseTimer* either on power up or after a Hard Reset. On entry to this state the Policy Engine **Shall** stop incrementing the SR_Timer.

The Policy Engine **Shall** transition to the **SRT_Running** State:

- When the *SenderResponseTimer* is started from within a Policy Engine state, or
- When a Start_SRT is requested from the Chunking Layer.

8.3.3.1.1.2 SRT_Running State

On entry to the **SRT_Running** State the *SenderResponseTimer* state machine **Shall**:

- Set the SR_Timer to zero
- Start running SR_Timer.

The *SenderResponseTimer* state machine **Shall** transition to the **SRT_Expired** State:

- When the SR_Timer reaches its maximum count

The ***SenderResponseTimer*** state machine ***Shall*** transition to the ***SRT_Stopped*** State:

- When the ***SenderResponseTimer*** is stopped by exiting a Policy Engine state, or
- When a Stop_SRT is requested from the Chunking Layer

8.3.3.1.1.3 SRT_Expired State

On entry to the ***SRT_Running*** State the ***SenderResponseTimer*** state machine ***Shall*** Inform Policy Engine of ***SenderResponseTimer*** timeout

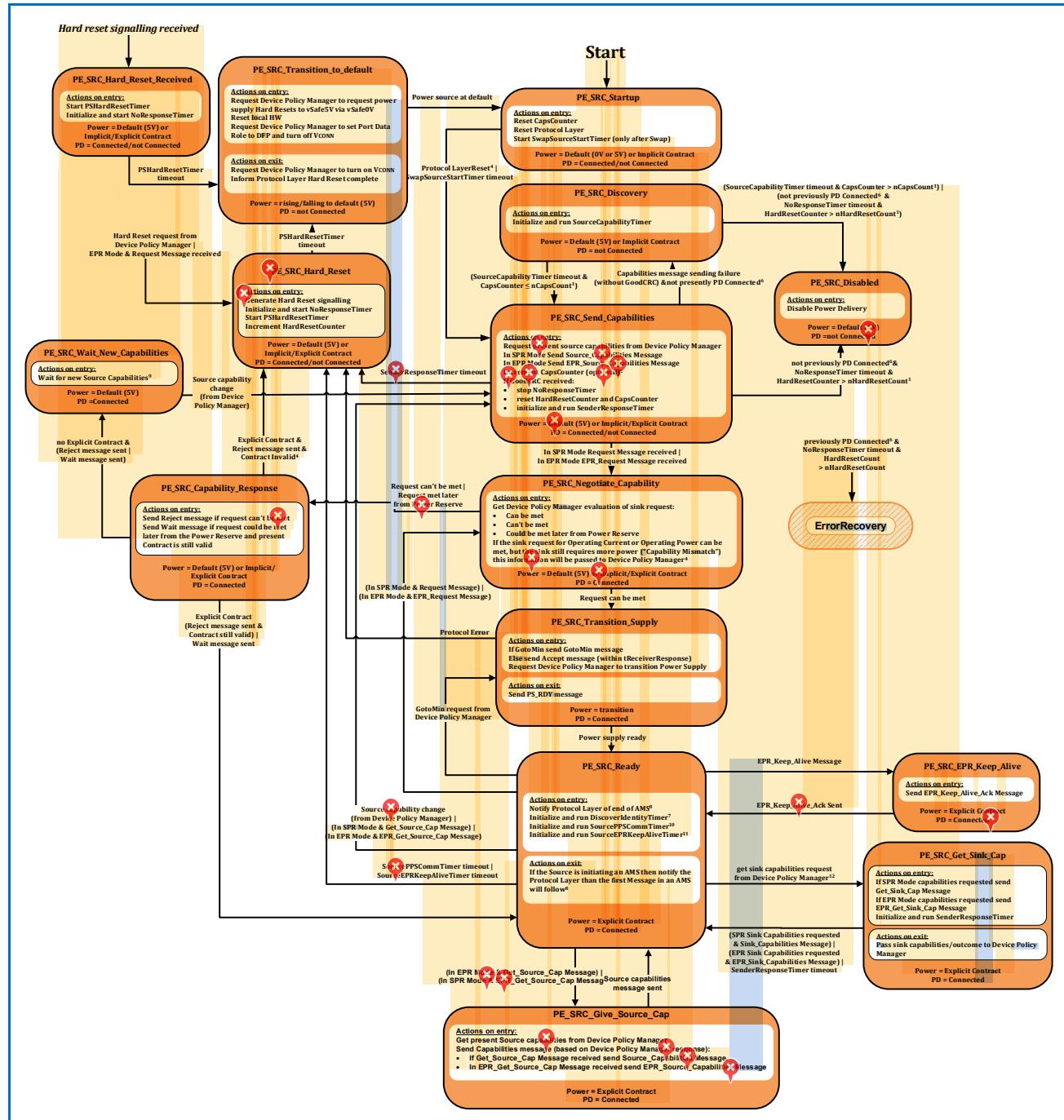
The Policy Engine ***Shall*** then transition to the ***SRT_Stopped*** state:

- When the Policy Engine has been informed.

8.3.3.2 Policy Engine Source Port State Diagram

Figure 8-134 “Source Port State Diagram” below shows the state diagram for the Policy Engine in a Source Port. The following sections describe operation in each of the states.

Figure 8-134 “Source Port State Diagram”



- ① Implementation of the **CapsCounter** is **Optional**. In the case where this is not implemented the Source **Shall** continue to send **Source_Capabilities** Messages each time the **SourceCapabilityTimer** times out.
- ② Since the Sink is required to make a **Valid** request from the offered capabilities the expected transition is via "Request can be met" unless the Source capabilities have changed since the last offer.
- ③ "Contract **Invalid**" means that the previously negotiated Voltage and Current values are no longer included in the Source's new Capabilities. If the Sink fails to make a **Valid** Request in this case, then Power Delivery operation is no longer possible and Power Delivery mode is exited with a Hard Reset.
- ④ After a Power Swap the new Source is required to wait an additional **tSwapSourceStart** before sending a **Source_Capabilities** Message. This delay is not required when first starting up a system.
- ⑤ PD Connected is defined as a situation when the Port Partners are actively communicating. The Port Partners remain PD Connected after a Swap until there is a transition to Disabled or the connector is able to identify a Detach.
- ⑥ Port Partners are no longer PD Connected after a Hard Reset, but consideration needs to be given as to whether there has been a PD Connection while the Ports have been Attached to prevent unnecessary USB Type-C® Error Recovery.
- ⑦ The **DiscoverIdentityTimer** is run when this is a VCONN Source and a PD Connection with a Cable Plug needs to be established i.e. no **GoodCRC** Message has yet been received in response to a **Discover Identity** Command.
- ⑧ See [Section 5.7 "Collision Avoidance"](#), [Section 6.6.16 "Collision Avoidance Timers"](#) and [Section 6.10 "Collision Avoidance"](#).
- ⑨ In the **PE_SRC_Wait_New_Capabilities** State the Device Policy Manager **Should** either decide to send no further Source Capabilities or **Should** send a different set of Source Capabilities. Continuing to send the same set of Source Capabilities could result in a live lock situation.
- ⑩ The **SourcePPSCommTimer** is only initialized and run when the present Explicit Contract is for an SPR PPS APDO. Sources that do not support SPR PPS do not need to implement the **SourcePPSCommTimer**.
- ⑪ The **SourceEPRKeepAliveTimer** is only initialized and run when the Source is in EPR Mode; Sources that do not support EPR Mode do not need to implement the **SourceEPRKeepAliveTimer**.
- ⑫ Either SPR or EPR Sink Capabilities **May** be requested, regardless of whether or not the Source is currently operating in SPR or EPR Mode.

8.3.3.2.1 PE_SRC_Startup State

PE_SRC_Startup **Shall** be the starting state for a Source Policy Engine either on power up or after a Hard Reset. On entry to this state the Policy Engine **Shall** reset the **CapsCounter** and reset the Protocol Layer. Note that resetting the Protocol Layer will also reset the **MessageIDCounter** and stored **MessageID** (see [Section 6.12.2.3 "Protocol Layer Message Reception"](#)).

The Policy Engine **Shall** transition to the **PE_SRC_Send_Capabilities** state:

- When the Protocol Layer reset has completed if the **PE_SRC_Startup** state was entered due to the system first starting up.
- When the **SwapSourceStartTimer** times out if the **PE_SRC_Startup** state was entered as the result of a Power Role Swap.

Note: Sources **Shall** remain in the **PE_SRC_Startup** state, without sending any **Source_Capabilities** Messages until a plug is Attached.

8.3.3.2.2 PE_SRC_Discovery State

On entry to the **PE_SRC_Discovery** state the Policy Engine **Shall** initialize and run the **SourceCapabilityTimer** in order to trigger sending a **Source_Capabilities** Message.

The Policy Engine **Shall** transition to the **PE_SRC_Send_Capabilities** state when:

- The **SourceCapabilityTimer** times out and **CapsCounter** $\leq nCapsCount$.

The Policy Engine **May Optionally** go to the **PE_SRC_Disabled** state when:

- The Port Partners are not presently PD Connected
- And the **SourceCapabilityTimer** times out
- And **CapsCounter** $> nCapsCount$.

✖ The Policy Engine **Shall** go to the **PE_SRC_Disabled** state when:

- The Port Partners have not been PD Connected (the Source Port remains Attached to a Port it has not had a PD Connection with during this Attachment)
- And the **NoResponseTimer** times out
- And the **HardResetCounter** $> nHardResetCount$.

Note in the **PE_SRC_Disabled** state the Attached device is assumed to be unresponsive. The Policy Engine operates as if the device is Detached until such time as a Detach/re-Attach is detected.

8.3.3.2.3 PE_SRC_Send_Capabilities State

Note: this state can be entered from the **PE_SRC_Soft_Reset** state.

On entry to the **PE_SRC_Send_Capabilities** state the Policy Engine **Shall** request the present Port capabilities from the Device Policy Manager. The Policy Engine **Shall** then request the Protocol Layer to send a capabilities message containing these capabilities. The Policy Engine **Shall** request:

- A **Source_Capabilities** Message if the Source is in SPR Mode or
- An **EPR_Source_Capabilities** Message if the Source is in EPR Mode.

The Policy Engine **Shall** then increment the **CapsCounter** (if implemented).

If a **GoodCRC** Message is received, then the Policy Engine **Shall**:

- Stop the **NoResponseTimer**.
- Reset the **HardResetCounter** and **CapsCounter** to zero. Note that the **HardResetCounter** **Shall** only be set to zero in this state and at power up; its value **Shall** be maintained during a Hard Reset.
- Initialize and run the **SenderResponseTimer**.

Once a **Source_Capabilities** Message has been received and acknowledged by a **GoodCRC** Message, the Sink is required to then send a **Request** Message within **tSenderResponse**.

The Policy Engine **Shall** transition to the **PE_SRC_Negotiate_Capability** state when:

- A **Request** Message is received from the Sink and the Source is operating in SPR Mode or
- An **EPR_Request** Message is received from the Sink and the Source is operating in EPR Mode.

The Policy Engine **Shall** transition to the **PE_SRC_Discovery** state when:

- The Protocol Layer indicates that the Message has not been sent and we are presently not Connected. This is part of the Capabilities sending process whereby successful Message sending indicates connection to a PD Sink Port.

The Policy Engine **Shall** transition to the **PE_SRC_Hard_Reset** state when:

- The **SenderResponseTimer** times out. In this case a transition back to USB Default Operation is required.

When:

- The Port Partners have not been PD Connected (the Source Port remains Attached to a Port it has not had a PD Connection with during this Attachment)
- And the **NoResponseTimer** times out
- And the **HardResetCounter > nHardResetCount**.

The Policy Engine **Shall** do one of the following:

- Transition to the **PE_SRC_Discovery** state.
- Transition to the **PE_SRC_Disabled** state.

 Note that in either case the Attached device is assumed to be unresponsive. The Policy Engine **Should** operate as if the device is Detached until such time as a Detach/re-Attach is detected.

The Policy Engine **Shall** go to the **ErrorRecovery** state when:

- The Port Partners have previously been PD Connected (the Source Port remains Attached to a Port it has had a PD Connection with during this Attachment)
- And the **NoResponseTimer** times out.
- And the **HardResetCounter > nHardResetCount**.

8.3.3.2.4 PE_SRC_Negotiate_Capability State

On entry to the **PE_SRC_Negotiate_Capability** state the Policy Engine **Shall** ask the Device Policy Manager to evaluate the Request from the Attached Sink. The response from the Device Policy Manager **Shall** be one of the following:

- The Request can be met.
- The Request cannot be met
- The Request could be met later from the Power Reserve.

The Policy Engine **Shall** transition to the **PE_SRC_Transition_Supply** state when:

- The Request can be met.

The Policy Engine **Shall** transition to the **PE_SRC_Capability_Response** state when:

- The Request cannot be met.
- Or the Request can be met later from the Power Reserve.

8.3.3.2.5 PE_SRC_Transition_Supply State

The Policy Engine **Shall** be in the **PE_SRC_Transition_Supply** state while the power supply is transitioning from one power to another.

On entry to the **PE_SRC_Transition_Supply** state, the Policy Engine **Shall** request the Protocol Layer to either send a **GotoMin** Message (if this was requested by the Device Policy Manager) or otherwise an **Accept** Message and inform

the Device Policy Manager that it ***Shall*** transition the power supply to the Requested power level. Note: that if the power supply is currently operating at the requested power no change will be necessary.

On exit from the ***PE_SRC_Transition_Supply*** state the Policy Engine ***Shall*** request the Protocol Layer to send a ***PS_RDY*** Message.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Ready*** state when:

- The Device Policy Manager informs the Policy Engine that the power supply is ready.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Hard_Reset*** state when:

- A Protocol Error occurs.

8.3.3.2.6 PE_SRC_Ready State

In the ***PE_SRC_Ready*** state the PD Source ***Shall*** be operating at a stable power with no ongoing negotiation. It ***Shall*** respond to requests from the Sink, events from the Device Policy Manager.

On entry to the ***PE_SRC_Ready*** state the Source ***Shall*** notify the Protocol Layer of the end of the Atomic Message Sequence (AMS). If the transition into ***PE_SRC_Ready*** is the result of Protocol Error that has not caused a Soft Reset (see [Section 8.3.3.4.1 "SOP Source Port Soft Reset and Protocol Error State Diagram"](#)) then the notification to the Protocol Layer of the end of the AMS ***Shall Not*** be sent since there is a Message to be processed.

On entry to the ***PE_SRC_Ready*** state if this is a VCONN Source which needs to establish communication with a Cable Plug, the Policy Engine ***Shall***:

- Initialize and run the ***DiscoverIdentityTimer*** (no ***GoodCRC*** Message response yet received to ***Discover Identity*** Message).

On entry to the ***PE_SRC_Ready*** state if the current Explicit Contract is for an SPR PPS APDO, then the Policy Engine ***Shall*** do the following:

- Initialize and run the ***SourcePPSCommTimer***.

On entry to the ***PE_SRC_Ready*** state if the current Explicit Contract is for EPR mode, then the Policy Engine ***Shall*** do the following:

- Initialize and run the ***SourceEPRKeepAliveTimer***.

On exit from the ***PE_SRC_Ready***, if the Source is initiating an AMS, then the Policy Engine ***Shall*** notify the Protocol Layer that the first Message in an AMS will follow.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Send_Capabilities*** state when:

- The Device Policy Manager indicates that Source Capabilities have changed or
- A ***Get_Source_Cap*** Message is received, and the Source is in SPR Mode or
- An ***EPR_Get_Source_Cap*** Message is received, and the Source is in EPR Mode.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Negotiate_Capability*** state when:

- A ***Request*** Message is received, and the Source is in SPR Mode or
- An ***EPR_Request*** Message is received, and the Source is in EPR Mode. *

The Policy Engine ***Shall*** transition to the ***PE_SRC_Transition_Supply*** state when:

- A GotoMin request is received from the Device Policy Manager for the Attached Device to go to minimum power.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Get_Sink_Cap*** state when:

- The Device Policy Manager asks for the Sink's capabilities.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Hard_Reset*** state when:

- The Source is operating as an SPR PPS and the ***SourcePPSCommTimer*** Timer times-out out or
- The Source is in EPR Mode and the ***SourceEPRKeepAliveTimer*** Timer times-out.

The Policy Engine ***Shall*** transition to the ***PE_SRC_EPR_Keep_Alive*** state when:

- An ***EPR_KeepAlive*** Message is received.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Give_Source_Cap*** State when:

- In EPR Mode and a ***Get_Source_Cap*** Message is received or
- In SPR Mode and an ***EPR_Get_Source_Cap*** Message is received.

8.3.3.2.7 PE_SRC_Disabled State

In the ***PE_SRC_Disabled*** state the PD Source supplies default power and is unresponsive to USB Power Delivery messaging, but not to ***Hard Reset*** Signaling.

8.3.3.2.8 PE_SRC_Capability_Response State

The Policy Engine ***Shall*** enter the ***PE_SRC_Capability_Response*** state if there is a Request received from the Sink that cannot be met based on the present capabilities. When the present Contract is not within the present capabilities it is regarded as ***Invalid*** and a Hard Reset will be triggered.

On entry to the ***PE_SRC_Capability_Response*** state the Policy Engine ***Shall*** request the Protocol Layer to send one of the following:

- ***Reject*** Message – if the request cannot be met or the present Contract is ***Invalid***.
- ***Wait*** Message – if the request could be met later from the Power Reserve. A ***Wait*** Message ***Shall Not*** be sent if the present Contract is ***Invalid***.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Ready*** state when:

- There is an Explicit Contract and
- A ***Reject*** Message has been sent and the present Contract is still ***Valid*** or
- A ***Wait*** Message has been sent.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Hard_Reset*** state when:

- There is an Explicit Contract and
- The ***Reject*** Message has been sent and the present Contract is ***Invalid*** (i.e., the Sink had to request a new value so instead we will return to USB Default Operation).

The Policy Engine ***Shall*** transition to the ***PE_SRC_Wait_New_Capabilities*** state when:

- There is no Explicit Contract and
- A ***Reject*** Message has been sent or

- A *Wait* Message has been sent.

8.3.3.2.9 PE_SRC_Hard_Reset State

The Policy Engine ***Shall*** transition to the ***PE_SRC_Hard_Reset*** state from any state when:

- Hard Reset request from Device Policy Manager or
- In EPR Mode and
- A *Request* Message is received.

On entry to the ***PE_SRC_Hard_Reset*** state the Policy Engine ***Shall***:

- request the generation of ***Hard Reset*** Signaling by the PHY Layer
- initialize and run the ***NoResponseTimer***. Note that the ***NoResponseTimer*** ***Shall*** continue to run in every state until it is stopped or times out.
- initialize and run the ***PSHardResetTimer*** and increment the ***HardResetCounter***.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Transition_to_default*** state when:

- The ***PSHardResetTimer*** times out.

8.3.3.2.10 PE_SRC_Hard_Reset_Received State

The Policy Engine ***Shall*** transition from any state to the ***PE_SRC_Hard_Reset_Received*** state when:

- ***Hard Reset*** Signaling is detected.

On entry to the ***PE_SRC_Hard_Reset_Received*** state the Policy Engine ***Shall***:

- initialize and run the ***PSHardResetTimer***
- initialize and run the ***NoResponseTimer***. Note that the ***NoResponseTimer*** ***Shall*** continue to run in every state until it is stopped or times out.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Transition_to_default*** state when:

- The ***PSHardResetTimer*** times out.

8.3.3.2.11 PE_SRC_Transition_to_default State

On entry to the ***PE_SRC_Transition_to_default*** state the Policy Engine ***Shall***:

- indicate to the Device Policy Manager that the power supply ***Shall*** Hard Reset (see [Section 7.1.5 “Response to Hard Resets”](#))
- request a reset of the local hardware
- request the Device Policy Manager to set the Port Data Role to DFP and turn off VCONN.

On exit from the ***PE_SRC_Transition_to_default*** state the Policy Engine ***Shall***:

- request the Device Policy Manager to turn on VCONN
- inform the Protocol Layer that the Hard Reset is complete.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Startup*** state when:

- The Device Policy Manager indicates that the power supply has reached the default level.

8.3.3.2.12 PE_SRC_Get_Sink_Cap State

In this state the Policy Engine, due to a request from the Device Policy Manager, **Shall** request the capabilities from the Attached Sink.

- On entry to the **PE_SRC_Get_Sink_Cap** state the Policy Engine **Shall** request the Protocol Layer to send a get Sink Capabilities message in order to retrieve the Sink's capabilities. The Policy Engine **Shall** send:
- A **Get_Sink_Cap** Message when the Device Policy Manager requests SPR capabilities or
- An **EPR_Get_Sink_Cap** Message when the Device Policy Manager requests EPR Capabilities.

The Policy Engine **Shall** then start the **SenderResponseTimer**.

On exit from the **PE_SRC_Get_Sink_Cap** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (capabilities or response timeout).

The Policy Engine **Shall** transition to the **PE_SRC_Ready** state when:

- SPR Sink Capabilities were requested and a **Sink_Capabilities** Message is received or
- EPR Sink Capabilities were requested and an **EPR_Sink_Capabilities** Message is received or
- The **SenderResponseTimer** times out.

8.3.3.2.13 PE_SRC_Wait_New_Capabilities State

In this state the Policy Engine has been unable to negotiate an Explicit Contract and is waiting for new Capabilities from the Device Policy Manager.

The Policy Engine **Shall** transition to the **PE_SRC_Send_Capabilities** state when:

- The Device Policy Manager indicates that Source Capabilities have changed.

8.3.3.2.14 PE_SRC_EPR_Keep_Alive State

On entry to the **PE_SRC_EPR_Keep_Alive** State the Policy Engine **Shall** send a **EPR_KeepAlive_Ack** Message.

The Policy Engine **Shall** transition to the **PE_SRC_Ready** state when:

- The **EPR_KeepAlive_Ack** Message has been sent.

8.3.3.2.15 PE_SRC_Give_Source_Cap State

- On entry to the **PE_SRC_Give_Source_Cap** State the Policy Engine **Shall** request the Device Policy Manager for the current system capabilities.

The Policy Engine **Shall** then request the Protocol Layer to send a Source Capabilities Message containing these capabilities.

The Policy Engine **Shall** send:

- A **Source_Capabilities** Message when a **Get_Source_Cap** Message is received or
- An **EPR_Source_Capabilities** Message when a **EPR_Get_Source_Cap** Message is received.

The Policy Engine **Shall** transition to the **PE_SNK_Ready** state when:

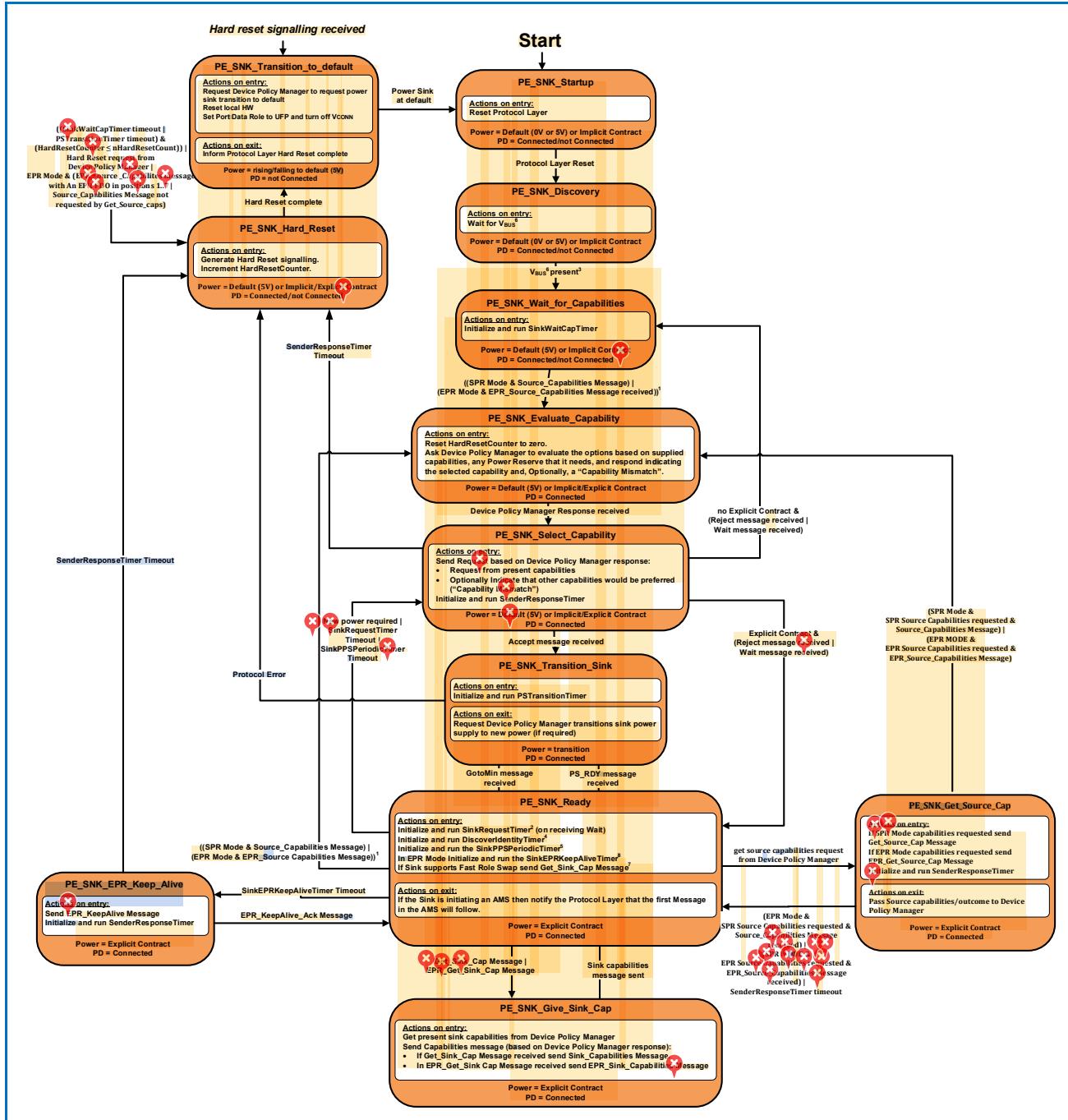
- The Source Capabilities Message has been successfully sent.

8.3.3.3

Policy Engine Sink Port State Diagram

Figure 8-135 "Sink Port State Diagram" below shows the state diagram for the Policy Engine in a Sink Port. The following sections describe operation in each of the states.

Figure 8-135 “Sink Port State Diagram”



- 1) Source capabilities messages received in States other than *PE_SNK_Wait_for_Capabilities*, *PE_SNK_Ready* or *PE_SNK_Get_Source_Cap* constitute a Protocol Error.
-    The *SinkRequestTimer* **Should Not** be stopped if a *Ping* Message is received in the *PE_SNK_Ready* state since it represents the maximum time between requests after a *Wait* Message which is not reset by a *Ping* Message.
- 3) During a Hard Reset the Source Voltage will transition to *vSafe0V* and then transition to *vSafe5V*. Sinks need to ensure that V_{BUS} present is not indicated until after the Source has completed the Hard-Reset process by detecting both of these transitions.
- 4) The *DiscoverIdentityTimer* is run when this is a VCONN Source and a PD Connection with a Cable Plug needs to be established i.e. no *GoodCRC* Message has yet been received in response to a *Discover Identity* Command.
- 5) The *SinkPPSPeriodicTimer* is only initialized and run when the present Explicit Contract is for an SPR PPS APDO. Sink's that do not support PPS do not need to implement the *SinkPPSPeriodicTimer*.
- 6) A Sink that is a VPD *May* use VCONN as a proxy for V_{BUS} .
- 7) To be sent once, and only required if Fast Role Swap is supported by the Sink.

8.3.3.1 PE_SNK_Startup State

PE_SNK_Startup **Shall** be the starting state for a Sink Policy Engine either on power up or after a Hard Reset. On entry to this state the Policy Engine **Shall** reset the Protocol Layer. Note that resetting the Protocol Layer will also reset the *MessageIDCounter* and stored *MessageID* (see *Section 6.12.2.3 "Protocol Layer Message Reception"*).

Once the reset process completes, the Policy Engine **Shall** transition to the *PE_SNK_Discovery* state.

8.3.3.2 PE_SNK_Discovery State

In the *PE_SNK_Discovery* state the Sink Policy Engine waits for V_{BUS} to be present.

The Policy Engine **Shall** transition to the *PE_SNK_Wait_for_Capabilities* state when:

- The Device Policy Manager indicates that V_{BUS} has been detected.

8.3.3.3 PE_SNK_Wait_for_Capabilities State

On entry to the *PE_SNK_Wait_for_Capabilities* state the Policy Engine **Shall** initialize and start the *SinkWaitCapTimer*.

The Policy Engine **Shall** transition to the *PE_SNK_Evaluate_Capability* state when:

- The Sink is in SPR Mode and a *Source_Capabilities* Message is received or
- The Sink is in EPR Mode and an *EPR_Source_Capabilities* Message is received.

When the *SinkWaitCapTimer* times out, the Policy Engine will perform a Hard Reset.

8.3.3.4 PE_SNK_Evaluate_Capability State

The *PE_SNK_Evaluate_Capability* state is first entered when the Sink receives its first *Source_Capabilities* Message from the Source. At this point the Sink knows that it is Attached to and communicating with a PD capable Source.

On entry to the *PE_SNK_Evaluate_Capability* state the Policy Engine **Shall** request the Device Policy Manager to evaluate the supplied Source capabilities based on Local Policy. The Device Policy Manager **Shall** indicate to the Policy Engine the new power level required, selected from the present offered capabilities. The Device Policy

Manager **Shall** also indicate to the Policy engine a Capability Mismatch if the offered power does not meet the device's requirements.

The Policy Engine **Shall** transition to the **PE_SNK_Select_Capability** state when:

- A response is received from the Device Policy Manager.

8.3.3.3.5 PE_SNK_Select_Capability State

On entry to the **PE_SNK_Select_Capability** state the Policy Engine **Shall** request the Protocol Layer to send a response Message, based on the evaluation from the Device Policy Manager. The Message **Shall** be one of the following:

- A Request from the offered Source Capabilities.
- A Request from the offered Source Capabilities with an indication that another power level would be preferred ("Capability Mismatch" bit set).

When in SPR Mode a **Request** Message **Shall** be sent.

When in EPR Mode an **EPR_Request** Message **Shall** be sent.

The Policy Engine **Shall** initialize and run the **SenderResponseTimer**.

The Policy Engine **Shall** transition to the **PE_SNK_Transition_Sink** state when:

- An **Accept** Message is received from the Source.

The Policy Engine **Shall** transition to the **PE_SNK_Wait_for_Capabilities** state when:

- There is no Explicit Contract in place and
- A **Reject** Message is received from the Source or
- A **Wait** Message is received from the Source.

The Policy Engine **Shall** transition to the **PE_SNK_Ready** state when:

- There is an Explicit Contract in place and
- A **Reject** Message is received from the Source or
- A **Wait** Message is received from the Source.

The Policy Engine **Shall** transition to the **PE_SNK_Hard_Reset** state when:

- A **SenderResponseTimer** timeout occurs.

8.3.3.6 PE_SNK_Transition_Sink State

On entry to the **PE_SNK_Transition_Sink** state the Policy Engine **Shall** initialize and run the **PSTransitionTimer** (timeout will lead to a Hard Reset see [Section 8.3.3.3.8 "PE_SNK_Hard_Reset State"](#) and **Shall** then request the Device Policy Manager to transition the Sink's power supply to the new power level. Note that if there is no power level change the Device Policy Manager **Should Not** affect any change to the power supply.

On exit from the **PE_SNK_Transition_Sink** state the Policy Engine **Shall** request the Device Policy Manager to transition the Sink's power supply to the new power level.

The Policy Engine **Shall** transition to the **PE_SNK_Ready** state when:

- A **PS_RDY** Message is received from the Source.

The Policy Engine **Shall** transition to the **PE_SNK_Hard_Reset** state when:

- A Protocol Error occurs.

8.3.3.7 PE_SNK_Ready State

In the **PE_SNK_Ready** state the PD Sink **Shall** be operating at a stable power level with no ongoing negotiation. It **Shall** respond to requests from the Source, events from the Device Policy Manager and **May** monitor for **Ping** Messages to maintain the PD link.

On entry to the **PE_SNK_Ready** state as the result of a wait the Policy Engine **Should** do the following:

- Initialize and run the **SinkRequestTimer**.

On entry to the **PE_SNK_Ready** state if this is a VCONN Source which needs to establish communication with a Cable Plug, then the Policy Engine **Shall** do the following:

- Initialize and run the **DiscoverIdentityTimer** (no **GoodCRC** Message response yet received to **Discover Identity** Message).

On entry to the **PE_SNK_Ready** state if the current Explicit Contract is for an SPR PPS APDO, then the Policy Engine **Shall** do the following:

- Initialize and run the **SinkPPSPeriodicTimer**.

On entry to the **PE_SNK_Ready** state if the Sink supports Fast Role Swap, then the Policy Engine **Shall** do the following:

- Send a **Get_Sink_Cap** Message.

On exit from the **PE_SNK_Ready** state, if the transition is as a result of a DPM request to start a new Atomic Message Sequence (AMS) then the Policy Engine **Shall** notify the Protocol Layer that the first Message in an AMS will follow.

The Policy Engine **Shall** transition to the **PE_SNK_Evaluate_Capability** state when:

- In SPR mode and a **Source_Capabilities** Message is received or
- In EPR mode and an **EPR_Source_Capabilities** Message is received.

The Policy Engine **Shall** transition to the **PE_SNK_Select_Capability** state when:

- A new power level is requested by the Device Policy Manager or
- A **SinkRequestTimer** timeout occurs or
- A **SinkPPSPeriodicTimer** timeout occurs.

The Policy Engine **Shall** transition to the **PE_SNK_Transition_Sink** state when:

- A **GotoMin** Message is received.

The Policy Engine **Shall** transition to the **PE_SNK_Give_Sink_Cap** state when:

- **Get_Sink_Cap** Message is received or
- **EPR_Get_Sink_Cap** Message is received.

The Policy Engine **Shall** transition to the **PE_SNK_Get_Source_Cap** state when:

- The Device Policy Manager requests an update of the remote Source's capabilities.

The Policy Engine **Shall** transition to the **PE_SNK_EPR_Keep_Alive** state when:

- The **SinkEPRKeepAliveTimer** timeouts out.

8.3.3.3.8 PE_SNK_Hard_Reset State

The Policy Engine **Shall** transition to the **PE_SNK_Hard_Reset** state from any state when:

- (*PSTransitionTimer* times out) &
- (*HardResetCounter* \leq *nHardResetCount*) |
- Hard Reset request from Device Policy Manager or
- In EPR Mode and
 - o An *EPR_Source_Capabilities* Message is received with an EPR PDO in object positions 1...7 or
 - o A *Source_Capabilities* Message is received that has not been requested using a *Get_Source_Cap* Message.

The Policy Engine **May** transition to the **PE_SNK_Hard_Reset** state from any state when:

- *SinkWaitCapTimer* times out

Note: if the *SinkWaitCapTimer* times out and the *HardResetCounter* is greater than *nHardResetCount* the Sink **Shall** assume that the Source is non-responsive.

Note: The *HardResetCounter* is reset on a power cycle or Detach.

On entry to the **PE_SNK_Hard_Reset** state the Policy Engine **Shall** request the generation of **Hard Reset** Signaling by the PHY Layer and increment the *HardResetCounter*.

The Policy Engine **Shall** transition to the **PE_SNK_Transition_to_default** state when:

- The Hard Reset is complete.

8.3.3.3.9 PE_SNK_Transition_to_default State

The Policy Engine **Shall** transition from any state to **PE_SNK_Transition_to_default** state when:

- **Hard Reset** Signaling is detected.

When **Hard Reset** Signaling is received or transmitted then the Policy Engine **Shall** transition from any state to **PE_SNK_Transition_to_default**. This state can also be entered from the **PE_SNK_Hard_Reset** state.

On entry to the **PE_SNK_Transition_to_default** state the Policy Engine **Shall**:

- indicate to the Device Policy Manager that the Sink **Shall** transition to default
- request a reset of the local hardware
- request the Device Policy Manager that the Port Data Role is set to UFP.

The Policy Engine **Shall** transition to the **PE_SNK_Startup** state when:

- The Device Policy Manager indicates that the Sink has reached the default level.

8.3.3.3.10 PE_SNK_Give_Sink_Cap State

- On entry to the **PE_SNK_Give_Sink_Cap** state the Policy Engine **Shall** request the Device Policy Manager for the current system capabilities. The Policy Engine **Shall** then request the Protocol Layer to send a *Sink_Capabilities* Message containing these capabilities. The Policy Engine **Shall** send:
 - A *Sink_Capabilities* Message when a *Get_Sink_Cap* Message is received or

- An *EPR_Sink_Capabilities* Message when a *EPR_Get_Sink_Cap* Message is received.

The Policy Engine ***Shall*** transition to the *PE_SNK_Ready* state when:

- The Sink Capabilities Message has been successfully sent.

8.3.3.3.11 PE_SNK_EPR_Keep_Alive

On entry to the *PE_SNK_EPR_Keep_Alive* State the Policy Engine ***Shall*** send an *EPR_KeepAlive* Message and initialize and run the *SenderResponseTimer*.

The Policy Engine ***Shall*** transition to the *PE_SNK_Ready* state when:

- A *EPR_KeepAlive_Ack* Message is received.

The Policy Engine ***Shall*** transition to the *PE_SNK_Hard_Reset* state when:

- The *SenderResponseTimer* times out.

8.3.3.3.12 PE_SNK_Get_Source_Cap State

- On entry to the *PE_SNK_Get_Source_Cap* state the Policy Engine ***Shall*** request the Protocol Layer to send a get Source Capabilities message in order to retrieve the Source's capabilities. The Policy Engine ***Shall*** send:
- A *Get_Source_Cap* Message when the Device Policy Manager requests SPR capabilities or
- An *EPR_Get_Source_Cap* Message when the Device Policy Manager requests EPR Capabilities.

The Policy Engine ***Shall*** then start the *SenderResponseTimer*.

On exit from the *PE_SNK_Get_Source_Cap* State the Policy Engine ***Shall*** inform the Device Policy Manager of the outcome (capabilities or response timeout).

The Policy Engine ***Shall*** transition to the *PE_SNK_Ready* state when:

- In EPR Mode and SPR Source Capabilities were requested and a *Source_Capabilities* Message is received or
- In SPR Mode and EPR Source Capabilities were requested and an *EPR_Source_Capabilities* Message is received or
- The *SenderResponseTimer* times out.

The Policy Engine ***Shall*** transition to the *PE_SNK_Evaluate_Capability* State when:

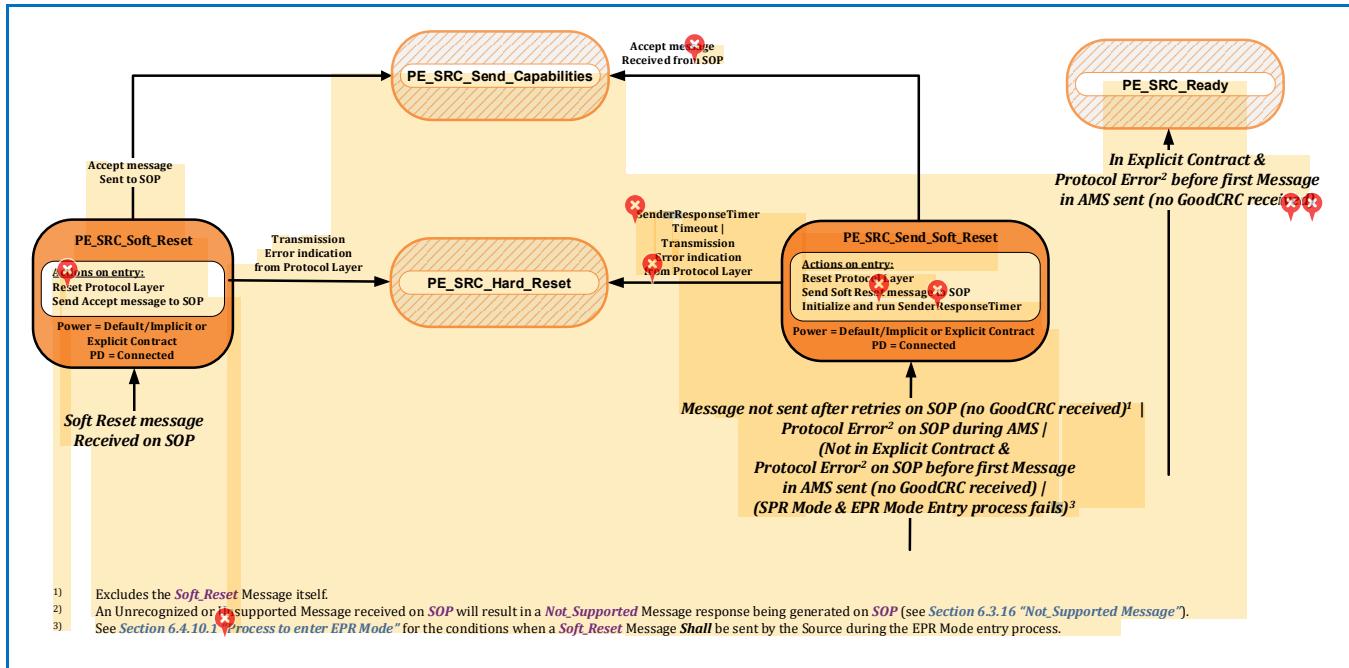
- In SPR Mode and SPR Source Capabilities were requested and a *Source_Capabilities* Message is received or
- In EPR Mode and EPR Source Capabilities were requested and an *EPR_Source_Capabilities* Message is received.

8.3.3.4 SOP Soft Reset and Protocol Error State Diagrams

8.3.3.4.1 SOP Source Port Soft Reset and Protocol Error State Diagram

Figure 8-136 “SOP Source Port Soft Reset and Protocol Error State Diagram” below shows the state diagram for the Policy Engine in a Source Port when performing a Soft Reset of its Port Partner i.e., using **SOP**. The following sections describe operation in each of the states.

Figure 8-136 “SOP Source Port Soft Reset and Protocol Error State Diagram”



8.3.3.4.1.1 PE_SRC_Send_Soft_Reset State

The **PE_SRC_Send_Soft_Reset** state **Shall** be entered from any state when:

- A Protocol Error on **SOP** is detected by the Protocol Layer during a Non-interruptible AMS (see [Section 8.8.1 “Soft Reset and Protocol Error”](#)) or
- A Message has not been sent after retries to the Sink or
- When not in an Explicit Contract and Protocol Errors occurred on **SOP** during any AMS where the first Message in the sequence has not yet been sent i.e., an unexpected Message is received instead of the expected **GoodCRC** Message response or
- When in SPR Mode and the EPR Mode entry process fails.

The main exceptions to this rule are when:

- The source is in the **PE_SRC_Send_Capabilities** state, there is a **Source_Capabilities** Message sending failure on **SOP** (without GoodCRC) and the source is not presently Attached (as indicated in [Figure 8-134 “Source Port State Diagram”](#)). In this case, the **PE_SRC_Discovery** state is entered (see [Section 8.3.3.2.2 “PE_SRC_Discovery State”](#)).
- When the Voltage is in transition due to a new Explicit Contract being negotiated (see [Section 8.3.3.2 “Policy Engine Source Port State Diagram”](#)). In this case Hard Reset Signaling will be generated.

- During a Power Role Swap when the power supply is in transition (see [Section 8.3.3.20.3 “Policy Engine in Source to Sink Power Role Swap State Diagram”](#) and [Section 8.3.3.20.4 “Policy Engine in Sink to Source Power Role Swap State Diagram”](#)). In this case USB Type-C® Error Recovery will be triggered directly.
- During a Data Role Swap when there is a mismatch in the Port Date Role field (see [Section 6.2.1.1.6 “Port Data Role”](#)). In this case USB Type-C® Error Recovery will be triggered directly.

Note that Protocol Errors occurring in the following situations ***Shall Not*** lead to a Soft Reset, but ***Shall*** result in a transition to the ***PE_SRC_Ready*** state where the Message received will be handled as if it had been received in the ***PE_SRC_Ready*** state:

- When in an Explicit Contract and Protocol Errors occurred on ***SOP*** during any AMS where the first Message in the sequence has not yet been sent i.e., an unexpected Message is received instead of the expected ***GoodCRC*** Message response.

On entry to the ***PE_SRC_Send_Soft_Reset*** state the Policy Engine ***Shall*** request the ***SOP*** Protocol Layer to perform a Soft Reset, then ***Shall*** send a ***Soft_Reset*** Message to the Sink on ***SOP***, and initialize and run the ***SenderResponseTimer***.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Send_Capabilities*** state when:

- An ***Accept*** Message has been received on ***SOP***.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Hard_Reset*** state when:

- A ***SenderResponseTimer*** timeout occurs.
- Or the Protocol Layer indicates that a transmission error has occurred.

8.3.3.4.1.2 PE_SRC_Soft_Reset State

The ***PE_SRC_Soft_Reset*** state ***Shall*** be entered from any state when a ***Soft_Reset*** Message is received on ***SOP*** from the Protocol Layer.

On entry to the ***PE_SRC_Soft_Reset*** state the Policy Engine ***Shall*** reset the ***SOP*** Protocol Layer and ***Shall*** then request the Protocol Layer to send an ***Accept*** Message on ***SOP***.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Send_Capabilities*** state (see [Section 8.3.3.2.3 “PE_SRC_Send_Capabilities State”](#)) when:

- The ***Accept*** Message has been sent on ***SOP***.

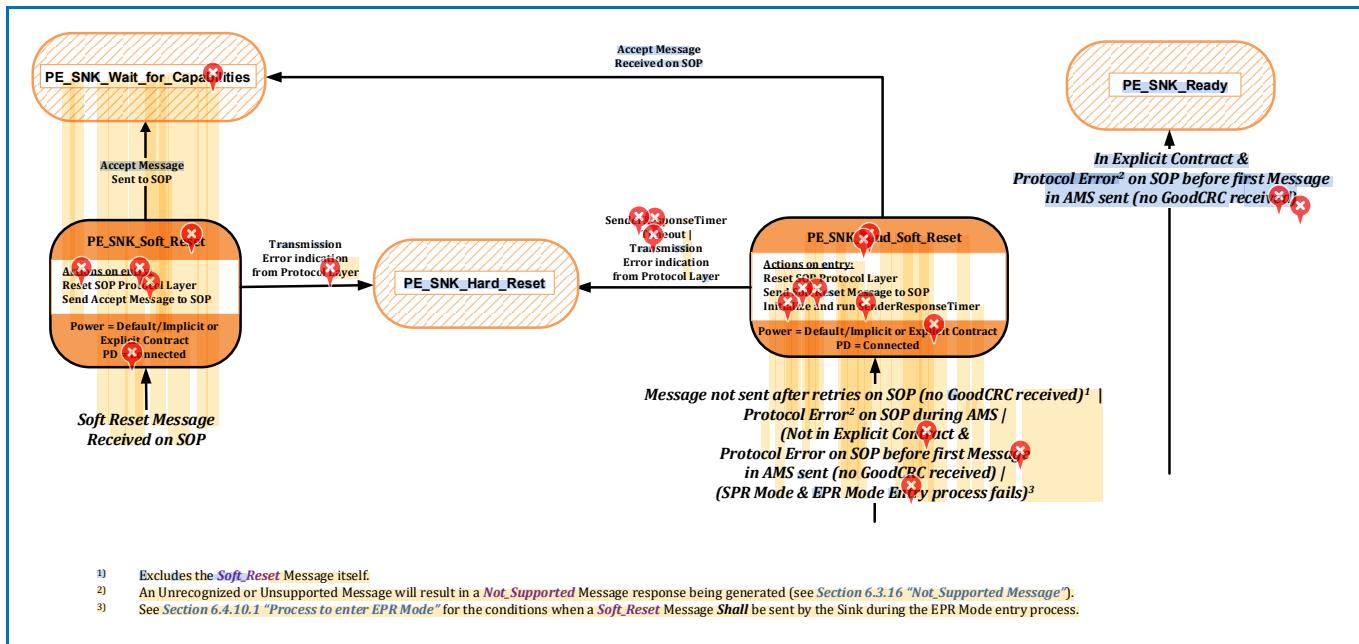
The Policy Engine ***Shall*** transition to the ***PE_SRC_Hard_Reset*** state when:

- The Protocol Layer indicates that a transmission error has occurred.

8.3.3.4.2 SOP Sink Port Soft Reset and Protocol Error State Diagram

Figure 8-137 “Sink Port Soft Reset and Protocol Error Diagram” below shows the state diagram for the Policy Engine in a Sink Port when performing a Soft Reset of its Port Partner i.e., using **SOP**. The following sections describe operation in each of the states.

Figure 8-137 “Sink Port Soft Reset and Protocol Error Diagram”



8.3.3.4.2.1 PE_SNK_Send_Soft_Reset State

The **PE_SNK_Send_Soft_Reset** state **Shall** be entered from any state when:

- A Protocol Error on **SOP** is detected by the Protocol Layer during an AMS (see [Section 6.8.1 “Soft Reset and Protocol Error”](#)) or
- A Message has not been sent after retries to the Sink or
- When not in an Explicit Contract and Protocol Errors occurred on **SOP** during any AMS where the first Message in the sequence has not yet been sent i.e., an unexpected Message is received instead of the expected **GoodCRC** Message response.
- When in SPR Mode and the EPR Mode entry process fails.

The main exceptions to this rule are when:

- When the Voltage is in transition due to a new Explicit Contract being negotiated (see [Section 8.3.3.3 “Policy Engine Sink Port State Diagram”](#)). In this case a Hard Reset will be generated.
- During a Power Role Swap when the power supply is in transition (see [Section 8.3.3.20.3 “Policy Engine in Source to Sink Power Role Swap State Diagram”](#) and [Section 8.3.3.20.4 “Policy Engine in Sink to Source Power Role Swap State Diagram”](#)). In this case a hard reset will be triggered directly.
- During a Data Role Swap when the DFP/UFP roles are changing. In this case USB Type-C® Error Recovery will be triggered directly.

Note that Protocol Errors occurring in the following situations ***Shall Not*** lead to a Soft Reset, but ***Shall*** result in a transition to the ***PE_SNK_Ready*** state where the Message received will be handled as if it had been received in the ***PE_SNK_Ready*** state:

- When in an Explicit Contract and Protocol Errors occurred on ***SOP*** during any AMS where the first Message in the sequence has not yet been sent i.e., an unexpected Message is received instead of the expected ***GoodCRC*** Message response.

On entry to the ***PE_SNK_Send_Soft_Reset*** state the Policy Engine ***Shall*** request the ***SOP*** Protocol Layer to perform a Soft Reset, then ***Shall*** send a ***Soft_Reset*** Message on ***SOP*** to the Source, and initialize and run the ***SenderResponseTimer***.

The Policy Engine ***Shall*** transition to the ***PE_SNK_Wait_for_Capabilities*** state when:

- An ***Accept*** Message has been received on ***SOP***.

The Policy Engine ***Shall*** transition to the ***PE_SNK_Hard_Reset*** state when:

- A ***SenderResponseTimer*** timeout occurs.
- Or the Protocol Layer indicates that a transmission error has occurred.

8.3.3.4.2.2 PE_SNK_Soft_Reset State

The ***PE_SNK_Soft_Reset*** state ***Shall*** be entered from any state when a ***Soft_Reset*** Message is received on ***SOP*** from the Protocol Layer.

On entry to the ***PE_SNK_Soft_Reset*** state the Policy Engine ***Shall*** reset the ***SOP*** Protocol Layer and ***Shall*** then request the Protocol Layer to send an ***Accept*** Message on ***SOP***.

The Policy Engine ***Shall*** transition to the ***PE_SNK_Wait_for_Capabilities*** state when:

- The ***Accept*** Message has been sent on ***SOP***.

The Policy Engine ***Shall*** transition to the ***PE_SNK_Hard_Reset*** state when:

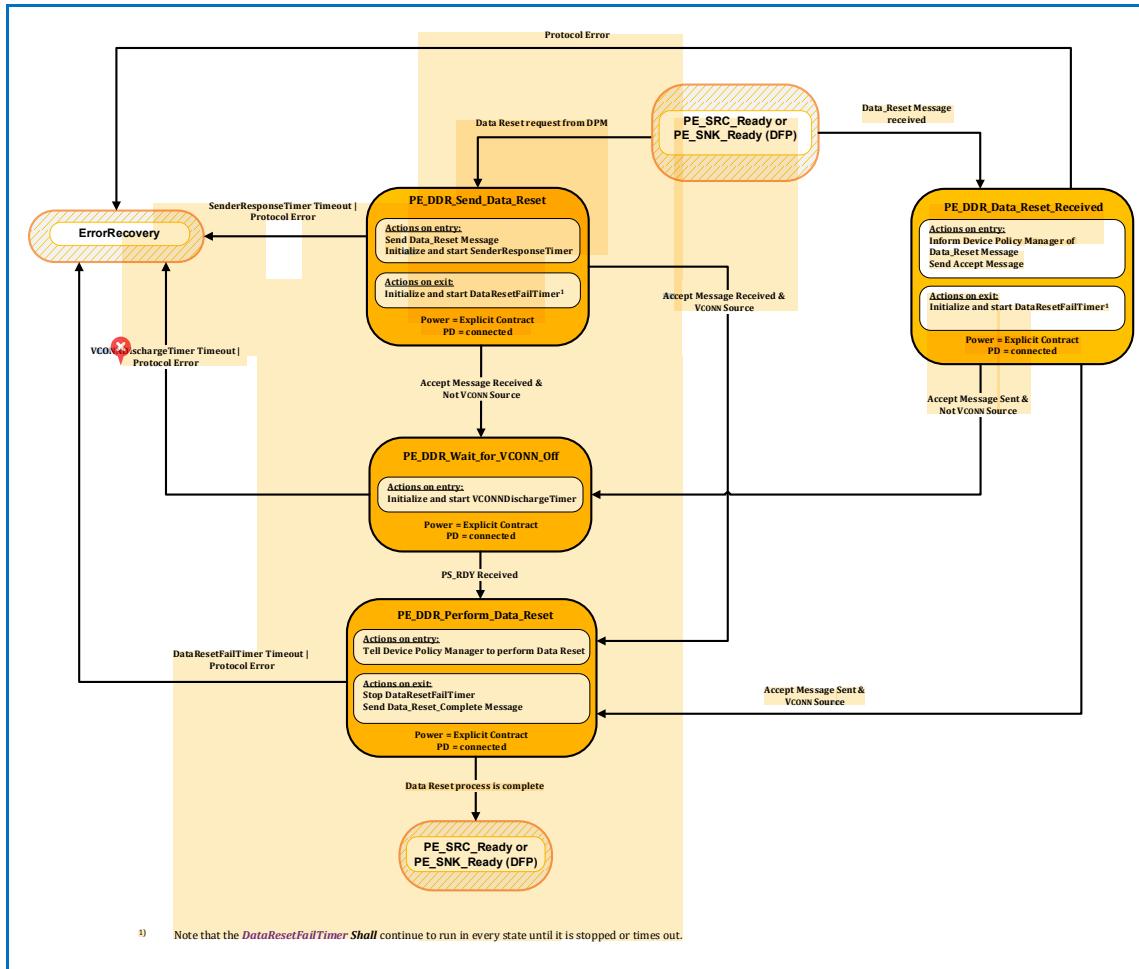
- The Protocol Layer indicates that a transmission error has occurred.

8.3.3.5 Data Reset State Diagrams

8.3.3.5.1 DFP Data_Reset Message State Diagrams

Figure 8-138 “DFP Data_Reset Message State Diagram” shows the state diagram for a **Data_Reset** Message sent or received by a DFP.

✖ **Figure 8-138 “DFP Data_Reset Message State Diagram”**



8.3.3.5.1.1 PE_DDR_Send_Data_Reset State

The **PE_DDR_Send_Data_Reset** State **Shall** be entered from the **PE_SRC_Ready** or **PE_SNK_Ready** State when requested by the Device Policy Manager.

On entry to the **PE_DDR_Send_Data_Reset** State the Policy Engine **Shall** request the Protocol Layer to send a **Data_Reset** Message and then initialize and start the **SenderResponseTimer**.

On exit from the **PE_DDR_Send_Data_Reset** State the Policy Engine **Shall** transition to the **PE_DDR_Perform_Data_Reset** State when:

- An **Accept** Message has been received and
- The DFP is presently the VCONN Source.

The Policy Engine ***Shall*** transition to the ***PE_DDR_Wait_For_VCONN_Off*** State when:

- An ***Accept*** Message has been received and
- ✖ The DFP is not presently the VCONN Source.

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- A ***SenderResponseTimer*** timeout occurs or
- A Protocol Error occurs.

8.3.3.5.1.2 PE_DDR_Data_Reset_Received State

The ***PE_DDR_Data_Reset_Received*** State ***Shall*** be entered from the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** State when a ***Data_Reset*** Message is received.

On entry to the ***PE_DDR_Data_Reset_Received*** State the Policy Engine ***Shall*** inform the Device Policy Manager and then ***Shall*** send an ***Accept*** Message.

On exit from the ***PE_DDR_Data_Reset_Received*** State the Policy Engine ***Shall*** initialize and start the ***DataResetFailTimer***.

The Policy Engine ***Shall*** transition to the ***PE_DDR_Perform_Data_Reset*** State when:

- An ***Accept*** Message has been sent and
- The DFP is presently the VCONN Source.

The Policy Engine ***Shall*** transition to the ***PE_DDR_Wait_For_VCONN_Off*** State when:

- An ***Accept*** Message has been sent and
- The DFP is not presently the VCONN Source.

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- A Protocol Error occurs.

8.3.3.5.1.3 PE_DDR_Wait_For_VCONN_Off State

On entry to the ***PE_DDR_Wait_For_VCONN_Off*** State the Policy Engine ***Shall*** initialize and start the ***VCONNDischargeTimer***.

The Policy Engine ***Shall*** transition to the ***PE_DDR_Perform_Data_Reset*** State when:

- A ***PS_RDY*** Message is received.

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- The ***VCONNDischargeTimer*** has timed out or
- A Protocol Error occurs.

8.3.3.5.1.4 PE_DDR_Perform_Data_Reset State

On entry to the ***PE_DDR_Perform_Data_Reset*** State the Policy Engine ***Shall*** request the Device Policy Manager to complete the Data Reset process as defined in ***Section 6.3.14 "Data_Reset Message"***.

On exit from the ***PE_DDR_Perform_Data_Reset*** State the Policy Engine ***Shall*** stop the ***DataResetFailTimer*** and send a ***Data_Reset_Complete*** Message.

The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** State depending on the DFP's Power Role when:

- The DPM indicates that Data Reset process is complete (see [Section 6.3.14 "Data_Reset Message"](#)).

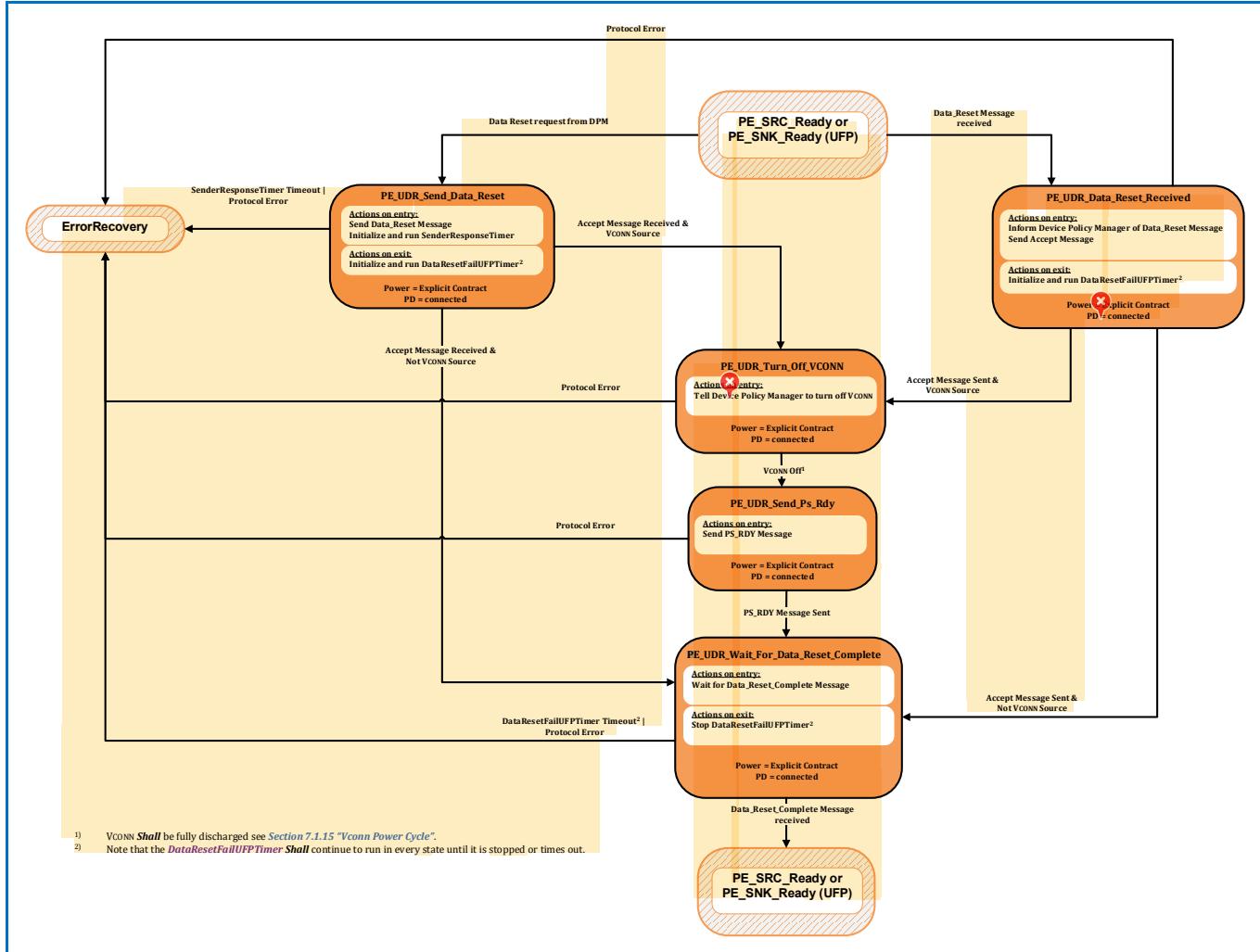
The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- The ***DataResetFailTimer*** times out
- A Protocol Error occurs.

8.3.3.5.2 UFP Data_Reset Message State Diagrams

Figure 8-139 “UFP Data_Reset Message State Diagram” shows the state diagram for a **Data_Reset** Message sent or received by a UFP.

Figure 8-139 “UFP Data_Reset Message State Diagram”



8.3.3.5.2.1 PE_UDR_Send_Data_Reset State

The **PE_UDR_Send_Data_Reset** State **Shall** be entered from the **PE_SRC_Ready** or **PE_SNK_Ready** State when requested by the Device Policy Manager.

On entry to the **PE_UDR_Send_Data_Reset** State the Policy Engine **Shall** request the Protocol Layer to send a **Data_Reset** Message and then initialize and run the **SenderResponseTimer**.

On exit from the **PE_UDR_Send_Data_Reset** State the Policy Engine **Shall** initialize and run the **DataResetFailUFPTimer**.

The Policy Engine **Shall** transition to the **PE_UDR_Turn_Off_VCONN** State when:

- An **Accept** Message has been received and
- The UFP is presently the VCONN Source.

The Policy Engine ***Shall*** transition to the ***PE_UDR_Wait_For_Data_Reset_Complete*** State when:

- An ***Accept*** Message has been received and 
- The UFP is not presently the VCONN Source.

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- The ***SenderResponseTimer*** has timed out or
- A Protocol Error occurs.

8.3.3.5.2.2 PE_UDR_Data_Reset_Received State

The ***PE_UDR_Data_Reset_Received*** State ***Shall*** be entered from either the ***PE_SRC_Rdy*** or ***PE_SNK_Rdy*** State when a ***Data_Reset*** Message is received.

On entry to the ***PE_UDR_Data_Reset_Received*** State the Policy Engine ***Shall*** inform the Device Policy Manager and then ***Shall*** send an ***Accept*** Message.

On exit from the ***PE_UDR_Data_Reset_Received*** State the Policy Engine ***Shall*** initialize and run the ***DataResetFailUFPTimer***.

The Policy Engine ***Shall*** transition to the ***PE_UDR_Turn_Off_VCONN*** State when:

- An ***Accept*** Message has been sent and
- The UFP is presently the VCONN Source.

The Policy Engine ***Shall*** transition to the ***PE_UDR_Wait_For_Data_Reset_Complete*** State when:

- An ***Accept*** Message has been sent and
- The UFP is not presently the VCONN Source.

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- A Protocol Error occurs.

8.3.3.5.2.3 PE_UDR_Turn_Off_VCONN State

On entry to the ***PE_UDR_Turn_Off_VCONN*** State the Policy Engine ***Shall*** request the Device Policy Manager to turn off VCONN.

The Policy Engine ***Shall*** transition to the ***PE_UDR_Send_Ps_Rdy*** State when:

- The DPM indicates that VCONN has been turned off (VCONN below vRaReconnect see **[USB Type-C 2.3]**).

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:

- A Protocol Error occurs.

8.3.3.5.2.4 PE_UDR_Send_Ps_Rdy State

On entry to the ***PE_UDR_Send_Ps_Rdy*** State the Policy Engine ***Shall*** send a ***PS_RDY*** Message.

The Policy Engine ***Shall*** transition to the ***PE_UDR_Wait_For_Data_Reset_Complete*** State when:

- The ***PS_RDY*** Message has been sent.

The Policy Engine ***Shall*** transition to ***ErrorRecovery*** when:



- A Protocol Error occurs.

8.3.3.5.2.5 PE_UDR_Wait_For_Data_Reset_Complete State

On entry to the **PE_UDR_Wait_For_Data_Reset_Complete** State the Policy Engine **Shall** wait for the **Data_Reset_Complete** Message.

On exit from the **PE_UDR_Wait_For_Data_Reset_Complete** State the Policy Engine **Shall** stop the **DataResetFailUFPTimer**.



The Policy Engine **Shall** transition back to either the **PE_SRC_Ready** or **PE_SNK_Ready** State depending on the UFP's Power Role when:

- The **Data_Reset_Complete** Message is received.

The Policy Engine **Shall** transition to **ErrorRecovery** when:

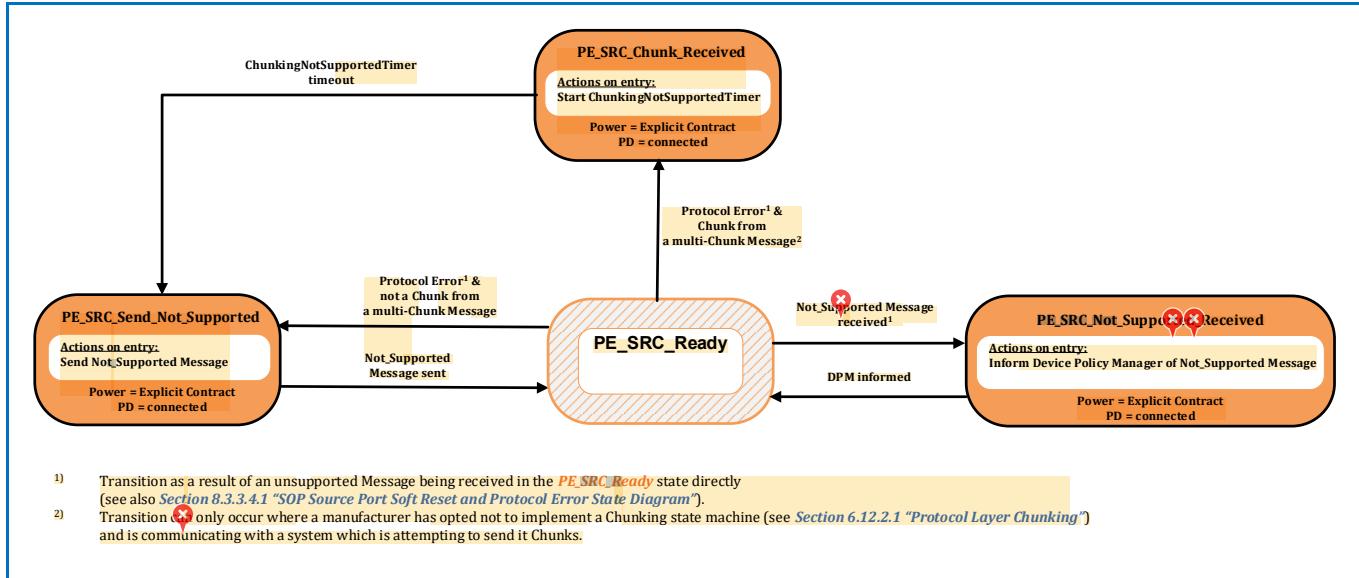
- The **DataResetFailUFPTimer** times out or
- A Protocol Error occurs.

8.3.3.6 Not Supported Message State Diagrams

8.3.3.6.1 Source Port Not Supported Message State Diagram

Figure 8-140 “Source Port Not Supported Message State Diagram” shows the state diagram for a *Not_Supported* Message sent or received by a Source Port.

Figure 8-140 “Source Port Not Supported Message State Diagram”



8.3.3.6.1.1 PE_SRC_Send_Not_Supported State

The **PE_SRC_Send_Not_Supported** state **Shall** be entered from the **PE_SRC_Ready** state either as the result of a Protocol Error received during an interruptible AMS or as a result of an unsupported Message being received in the **PE_SRC_Ready** state directly except for the first Chunk in a multi-Chunk Message (see also [Section 6.12.2.1 “Protocol Layer Chunking”](#) and [Section 8.3.3.4.1 “SOP Source Port Soft Reset and Protocol Error State Diagram”](#)).

On entry to the **PE_SRC_Send_Not_Supported** state (from the **PE_SRC_Ready** state) the Policy Engine **Shall** request the Protocol Layer to send a *Not_Supported* Message.

The Policy Engine **Shall** transition back to the previous state (**PE_SRC_Ready** see [Figure 8-134 “Source Port State Diagram”](#)) when:

- The *Not_Supported* Message has been successfully sent.

8.3.3.6.1.2 PE_SRC_Not_Supported_Received State

The **PE_SRC_Not_Supported_Received** state **Shall** be entered from the **PE_SRC_Ready** state when a *Not_Supported* Message is received.

On entry to the **PE_SRC_Not_Supported_Received** state the Policy Engine **Shall** inform the Device Policy Manager.

The Policy Engine **Shall** transition back to the previous state (**PE_SRC_Ready** see [Figure 8-134 “Source Port State Diagram”](#)) when:

- The Device Policy Manager has been informed.

8.3.3.6.1.3

PE_SRC_Chunk_Received State

✖ The **PE_SRC_Chunk_Received** state **Shall** be entered from the **PE_SRC_Ready** state as a result of an unsupported Message being received in the **PE_SRC_Ready** state directly where the Message is a Chunk in a multi-Chunk Message (see also [Section 6.12.2.1 “Protocol Layer Chunking”](#) and [Section 8.3.3.4.1 “SOP Source Port Soft Reset and Protocol Error State Diagram”](#)).

On entry to the **PE_SRC_Chunk_Received** state (from the **PE_SRC_Ready** state) the Policy Engine **Shall** initialize and run the **ChunkingNotSupportedTimer**.

The Policy Engine **Shall** transition to **PE_SRC_Send_Not_Supported** when:

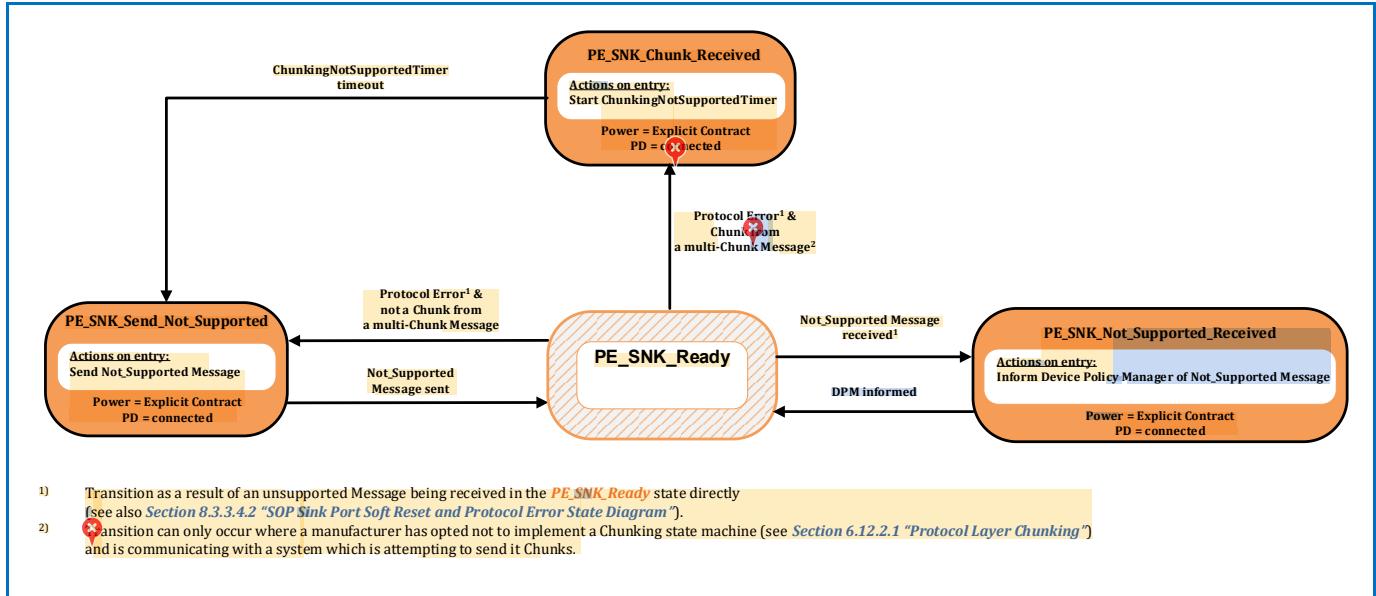
- The **ChunkingNotSupportedTimer** has timed out.

8.3.3.6.2

Sink Port Not Supported Message State Diagram

Figure 8-141 “Sink Port Not Supported Message State Diagram” shows the state diagram for a *Not_Supported* Message sent or received by a Sink Port.

Figure 8-141 “Sink Port Not Supported Message State Diagram”



8.3.3.6.2.1

PE_SNK_Send_Not_Supported State

The **PE_SNK_Send_Not_Supported** state **Shall** be entered from the **PE_SNK_Ready** state either as the result of a Protocol Error received during an interruptible AMS or as a result of an unsupported Message being received in the **PE_SNK_Ready** state directly except for the first Chunk in a multi-Chunk Message (see also [Section 6.12.2.1 “Protocol Layer Chunking”](#) and [Section 8.3.3.4.1 “SOP Source Port Soft Reset and Protocol Error State Diagram”](#)).

On entry to the **PE_SNK_Send_Not_Supported** state (from the **PE_SNK_Ready** state) the Policy Engine **Shall** request the Protocol Layer to send a *Not_Supported* Message.

The Policy Engine **Shall** transition back to the previous state (**PE_SNK_Ready** see [Figure 8-135 “Sink Port State Diagram”](#)) when:

- The *Not_Supported* Message has been successfully sent.

8.3.3.6.2.2

PE_SNK_Not_Supported_Received State

✖ The **PE_SNK_Not_Supported_Received** state **Shall** be entered from the **PE_SNK_Ready** state when a *Not_Supported* Message is received.

On entry to the **PE_SNK_Not_Supported_Received** state the Policy Engine **Shall** inform the Device Policy Manager.

The Policy Engine **Shall** transition back to the previous state (**PE_SNK_Ready** see [Figure 8-135 “Sink Port State Diagram”](#)) when:

- The Device Policy Manager has been informed.

8.3.3.6.2.3

PE_SNK_Chunk_Received State

The **PE_SNK_Chunk_Received** state **Shall** be entered from the **PE_SNK_Ready** state as a result of an unsupported Message being received in the **PE_SNK_Ready** state directly where the Message is a Chunk in a multi-Chunk Message (see also [Section 6.12.2.1 “Protocol Layer Chunking”](#) and [Section 8.3.3.4.1 “SOP Source Port Soft Reset and Protocol Error State Diagram”](#)).

On entry to the **PE_SNK_Chunk_Received** state (from the **PE_SNK_Ready** state) the Policy Engine **Shall** initialize and run the **ChunkingNotSupportedTimer**.

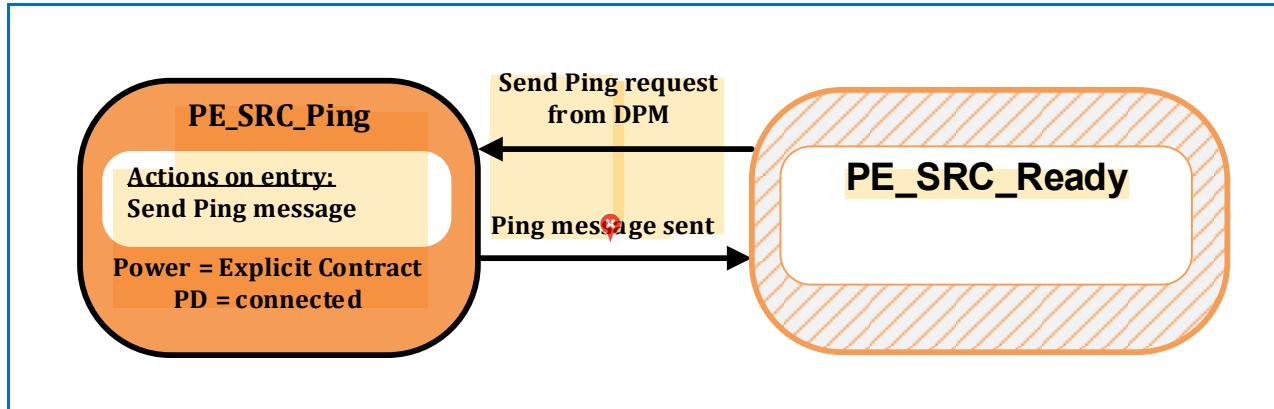
The Policy Engine **Shall** transition to **PE_SNK_Send_Not_Supported** when:

- The **ChunkingNotSupportedTimer** has timed out.

8.3.3.7 Source Port Ping State Diagram

Figure 8-142 “Source Port Ping State Diagram” shows the state diagram for a *Ping* Message from a Source Port.

Figure 8-142 “Source Port Ping State Diagram”



8.3.3.7.1 PE_SRC_Ping State

On entry to the **PE_SRC_Ping** state (from the **PE_SRC_Ready** state) the Policy Engine **Shall** request the Protocol Layer to send a *Ping* Message.

The Policy Engine **Shall** transition back to the previous state (**PE_SRC_Ready**) (see [Figure 8-134 “Source Port State Diagram”](#)) when:

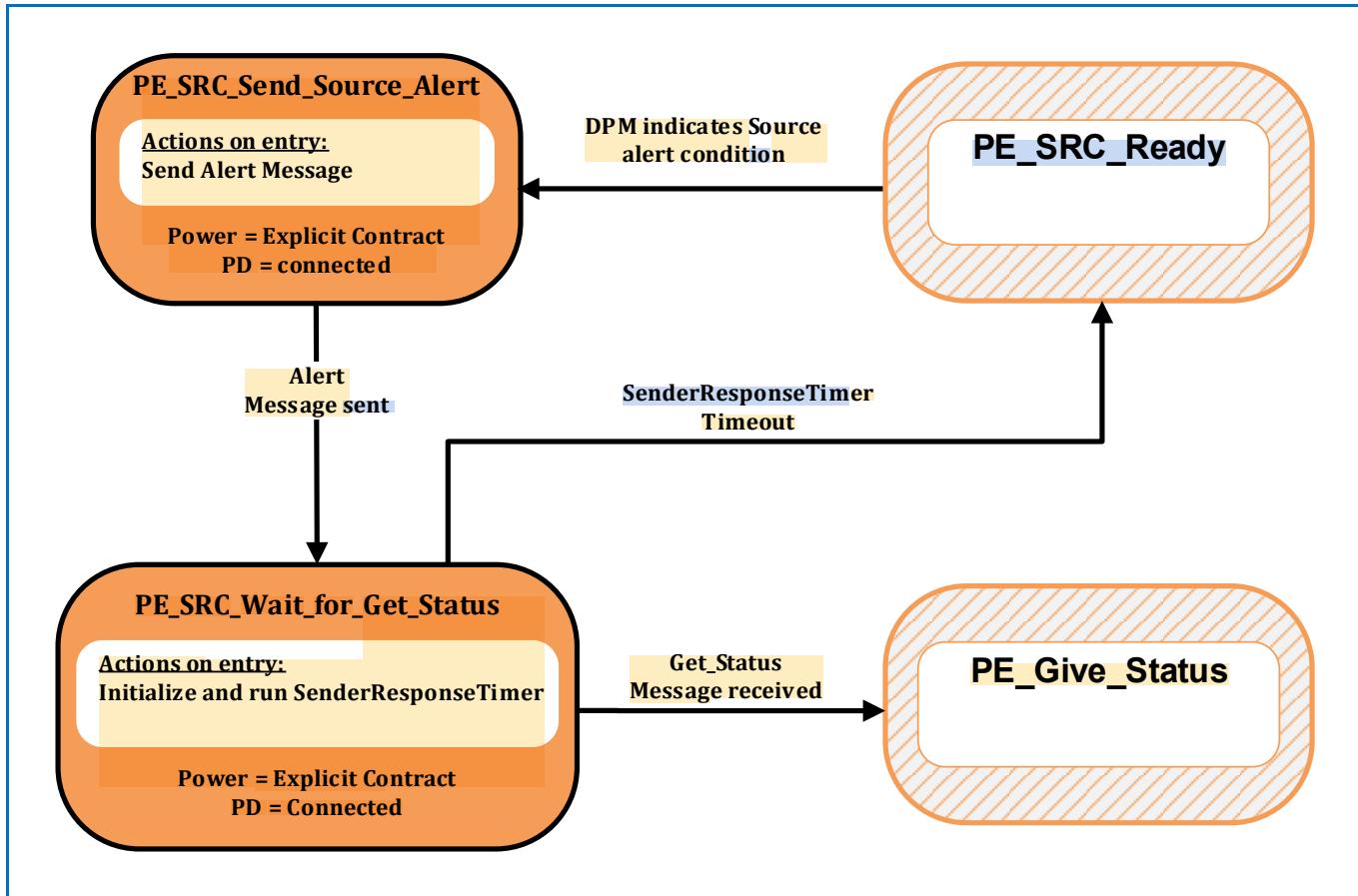
- The *Ping* Message has been successfully sent.

8.3.3.8 Alert State Diagrams

8.3.3.8.1 Source Port Source Alert State Diagram

Figure 8-143 “Source Port Source Alert State Diagram” shows the state diagram for an Alert Message sent by a Source Port.

Figure 8-143 “Source Port Source Alert State Diagram”



8.3.3.8.1.1 PE_SRC_Send_Source_Alert State

The **PE_SRC_Send_Source_Alert** state **Shall** be entered from the **PE_SRC_Ready** state when the Device Policy Manager indicates that there is a Source alert condition to be reported.

On entry to the **PE_SRC_Send_Source_Alert** state the Policy Engine **Shall** request the Protocol Layer to send an Alert Message.

The Policy Engine **Shall** transition to the **PE_SRC_Wait_for_Get_Status** State when:

- The **Alert** Message has been successfully sent.

8.3.3.8.1.2 PE_SRC_Wait_for_Get_Status State

On entry to the **PE_SRC_Wait_for_Get_Status** State the Policy Engine **Shall** initialize and run the **SenderResponseTimer**.

The Policy Engine ***Shall*** transition back to the ***PE_Give_Status*** State (see [Figure 8-154 “Give Status State Diagram”](#)) when:

- A ***Get_Status*** Message is received.

The Policy Engine ***Shall*** transition back to ***PE_SRC_Ready*** (see [Figure 8-134 “Source Port State Diagram”](#)) when:

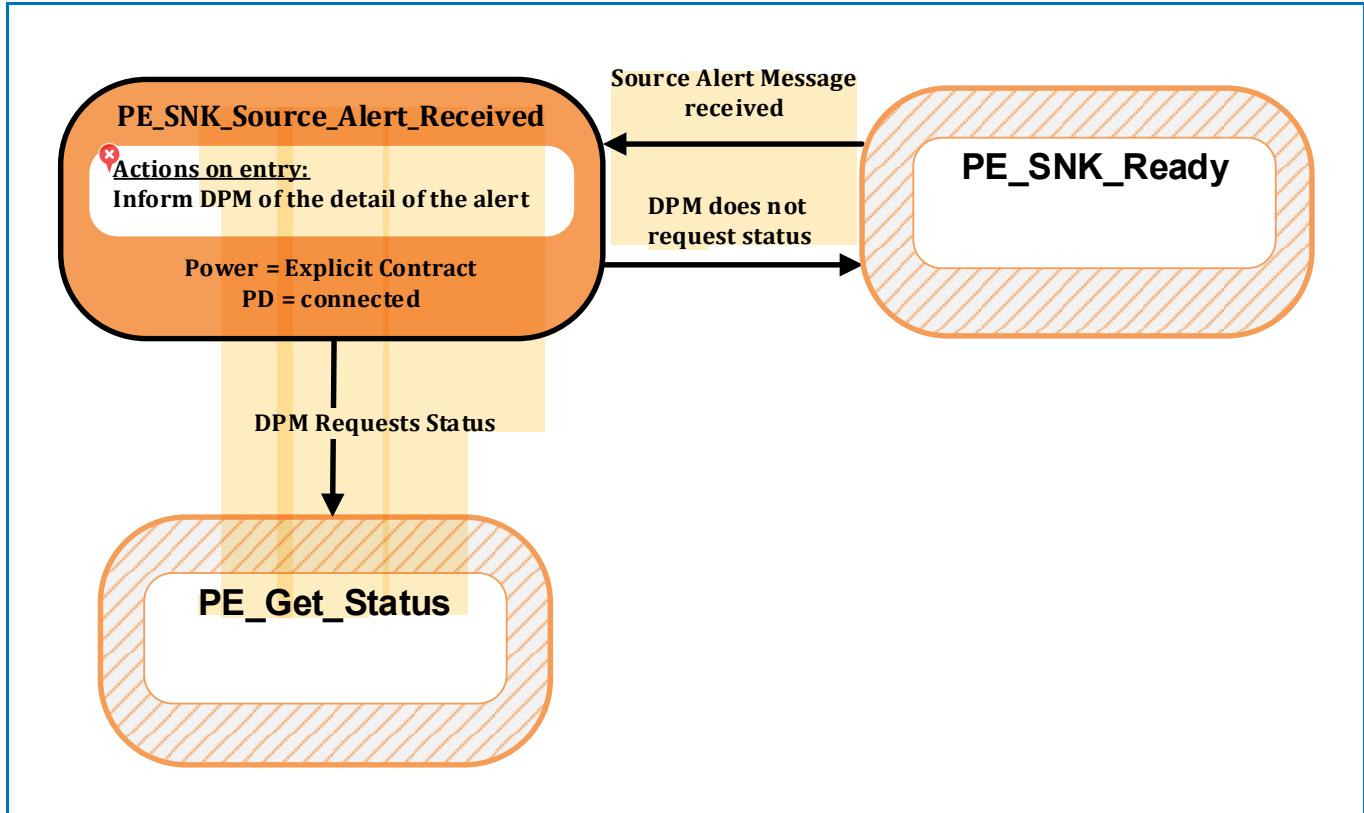
- The ***SenderResponseTimer*** times out.



8.3.3.8.2 Sink Port Source Alert State Diagram

[Figure 8-144 “Sink Port Source Alert State Diagram”](#) shows the state diagram for an Alert Message received by a Sink Port.

[Figure 8-144 “Sink Port Source Alert State Diagram”](#)



8.3.3.8.2.1 PE_SNK_Source_Alert_Received State

The **PE_SNK_Source_Alert_Received** state **Shall** be entered from the **PE_SNK_Ready** state when an Alert Message is received.

On entry to the **PE_SNK_Source_Alert_Received** state the Policy Engine **Shall** inform the Device Policy Manager of the details of the Source alert.

The Policy Engine **Shall** transition to the **PE_Get_Status** State (see [Figure 8-153 “Get Status State Diagram”](#)) when:

- The DPM requests status.

The Policy Engine **Shall** transition back to the **PE_SNK_Ready** State (see [Figure 8-135 “Sink Port State Diagram”](#)) when:

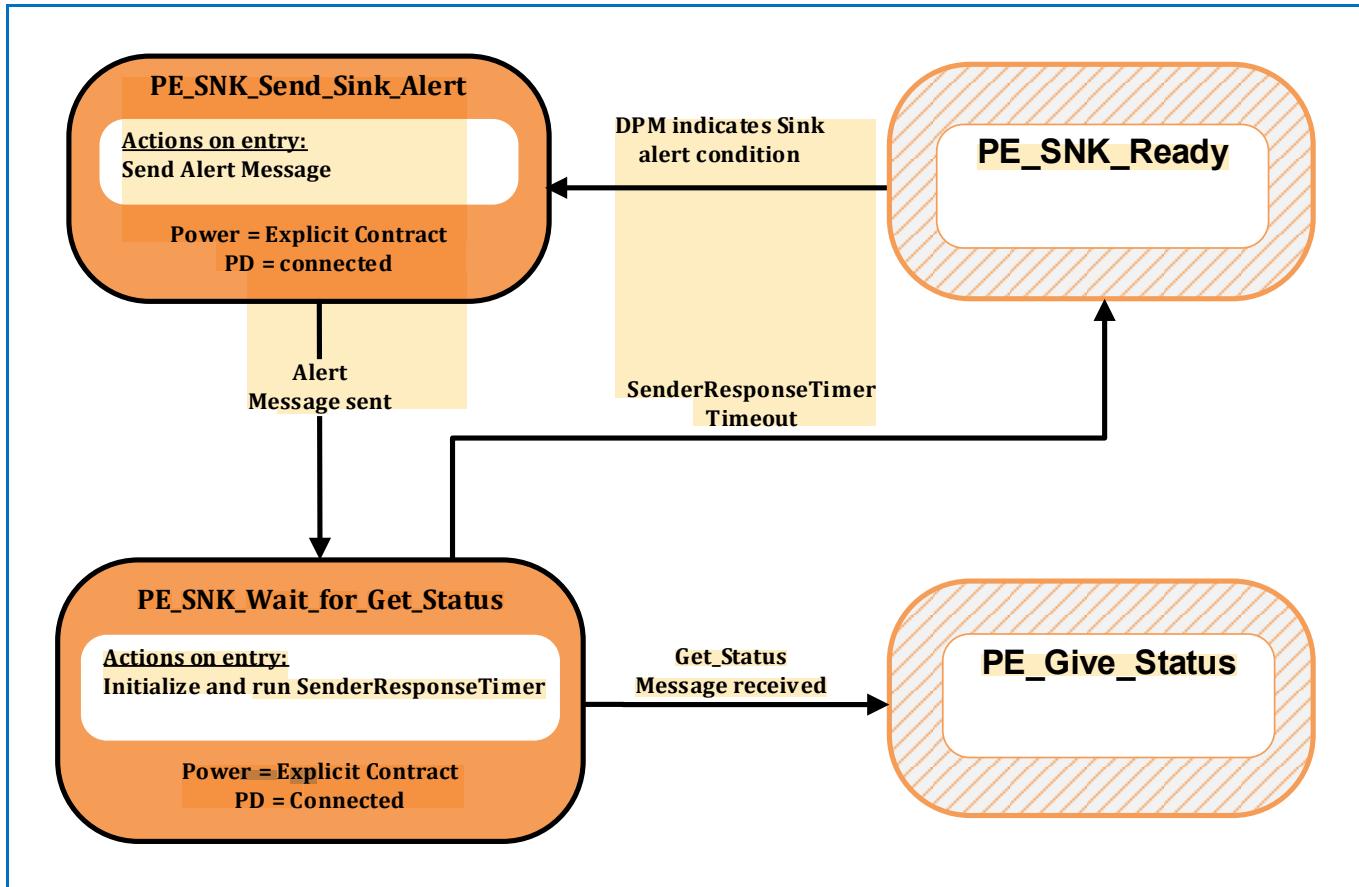
- The DPM does not request status.

8.3.3.8.3

Sink Port Sink Alert State Diagram

Figure 8-145 “Sink Port Sink Alert State Diagram” shows the state diagram for an Alert Message sent by a Sink Port.

Figure 8-145 “Sink Port Sink Alert State Diagram”



8.3.3.8.3.1

PE_SNK_Send_Sink_Alert State

The **PE_SNK_Send_Sink_Alert** state **Shall** be entered from the **PE_SNK_Ready** state when the Device Policy Manager indicates that there is a Source alert condition to be reported.

On entry to the **PE_SNK_Send_Sink_Alert** state the Policy Engine **Shall** request the Protocol Layer to send an Alert Message.

The Policy Engine **Shall** transition to the **PE_SNK_Wait_for_Get_Status** State when:

- The **Alert** Message has been successfully sent.

8.3.3.8.3.2

PE_SNK_Wait_for_Get_Status State

On entry to the **PE_SNK_Wait_for_Get_Status** State the Policy Engine **Shall** initialize and run the **SenderResponseTimer**.

The Policy Engine **Shall** transition back to the **PE_Give_Status** State (see **Figure 8-154 “Give Status State Diagram”**) when:

- A **Get_Status** Message is received.

The Policy Engine ***Shall*** transition back to the ***PE_SNK_Ready*** (see *Figure 8-135 “Sink Port State Diagram”*) when:

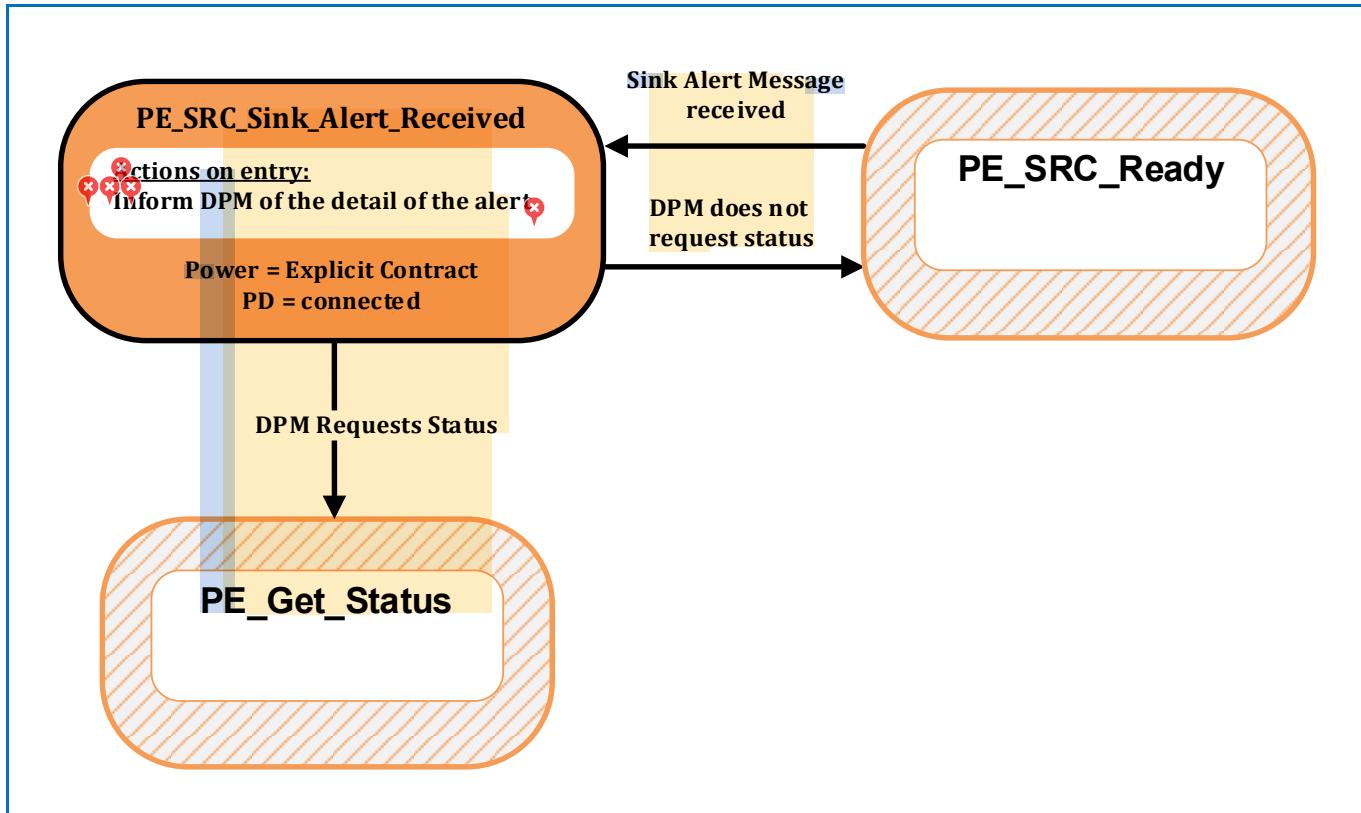
- The ***SenderResponseTimer*** times out.

8.3.3.8.4

Source Port Sink Alert State Diagram

Figure 8-146 “Source Port Sink Alert State Diagram” shows the state diagram for an Alert Message received by a Source Port.

Figure 8-146 “Source Port Sink Alert State Diagram”



8.3.3.8.4.1

PE_SRC_Sink_Alert_Received State

The **PE_SRC_Sink_Alert_Received** state **Shall** be entered from the **PE_SRC_Ready** state when an Alert Message is received.

On entry to the **PE_SRC_Sink_Alert_Received** state the Policy Engine **Shall** inform the Device Policy Manager of the details of the Source alert.

The Policy Engine **Shall** transition to the **PE_Get_Status** State (see Figure 8-153 “Get Status State Diagram”) when:

- The DPM requests status.

The Policy Engine **Shall** transition back to the **PE_SRC_Ready** (see Figure 8-134 “Source Port State Diagram”) when:

- The DPM does not request status.

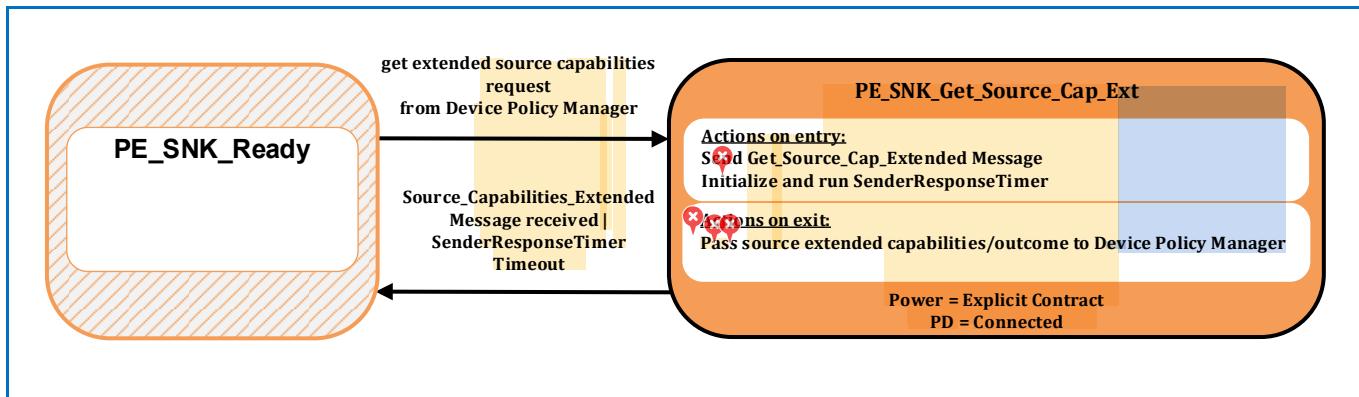


8.3.3.9 Source/Sink Capabilities Extended State Diagrams

8.3.3.9.1 Sink Port Get Source Capabilities Extended State Diagram

Figure 8-147 "Sink Port Get Source Capabilities Extended State Diagram" shows the state diagram for a Sink on receiving a request from the Device Policy Manager to get the Port Partner's extended Source capabilities. See also [Section 6.5.1 "Source_Capabilities_Extended Message"](#).

Figure 8-147 "Sink Port Get Source Capabilities Extended State Diagram"



8.3.3.9.1.1 PE_SNK_Get_Source_Cap_Ext State

The Policy Engine **Shall** transition to the **PE_SNK_Get_Source_Cap_Ext** state, from the **PE_SNK_Ready** state, due to a request to get the remote extended source capabilities from the Device Policy Manager.

On entry to the **PE_SNK_Get_Source_Cap_Ext** state the Policy Engine **Shall** send a **Get_Source_Cap_Extended** Message and initialize and run the **SenderResponseTimer**.

On exit from the **PE_SNK_Get_Source_Cap_Ext** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (capabilities or response timeout).

The Policy Engine **Shall** transition back to the **PE_SNK_Ready** state (see [Figure 8-135 "Sink Port State Diagram"](#)) when:

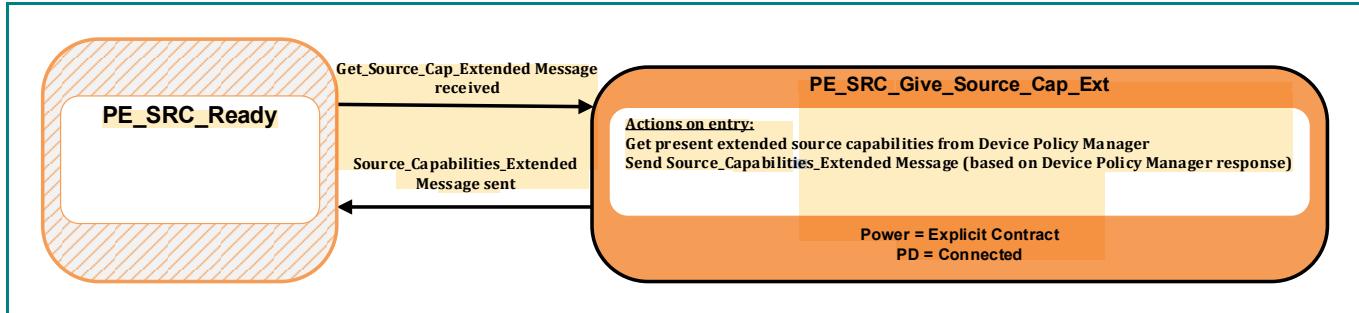
- A **Source_Capabilities_Extended** Message is received
- Or **SenderResponseTimer** times out.

8.3.3.9.2

Source Give Source Capabilities Extended State Diagram

Figure 8-148 “Source Give Source Capabilities Extended State Diagram” shows the state diagram for a Source on receiving a **Get_Source_Cap_Extended** Message. See also *Section 6.5.1 “Source_Capabilities_Extended Message”*.

Figure 8-148 “Source Give Source Capabilities Extended State Diagram”



8.3.3.9.2.1

PE_SRC_Give_Source_Cap_Ext State

The Policy Engine **Shall** transition to the **PE_SRC_Give_Source_Cap_Ext** state, from the **PE_SRC_Ready** state, when a **Get_Source_Cap_Extended** Message is received.

On entry to the **PE_SRC_Give_Source_Cap_Ext** state the Policy Engine **Shall** request the present extended Source capabilities from the Device Policy Manager and then send a **Source_Capabilities_Extended** Message based on these capabilities.

The Policy Engine **Shall** transition back to the **PE_SRC_Ready** state (see *Figure 8-134 “Source Port State Diagram”*) when:

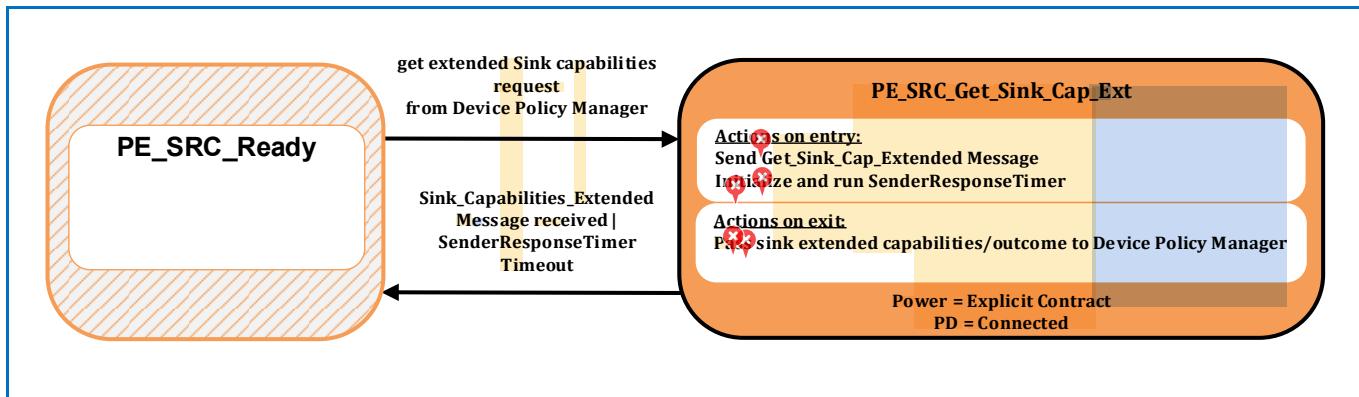
- The **Source_Capabilities_Extended** Message has been successfully sent.

8.3.3.9.3

Source Port Get Sink Capabilities Extended State Diagram

Figure 8-149 “Source Port Get Sink Capabilities Extended State Diagram” shows the state diagram for a Source on receiving a request from the Device Policy Manager to get the Port Partner’s extended Sink capabilities. See also [Section 6.5.13 “Sink_Capabilities_Extended Message”](#).

Figure 8-149 “Source Port Get Sink Capabilities Extended State Diagram”



8.3.3.9.3.1

PE_SRC_Get_Sink_Cap_Ext State

The Policy Engine **Shall** transition to the **PE_SRC_Get_Sink_Cap_Ext** state, from the **PE_SRC_Ready** state, due to a request to get the remote extended source capabilities from the Device Policy Manager.

On entry to the **PE_SRC_Get_Sink_Cap_Ext** state the Policy Engine **Shall** send a **Get_Sink_Cap_Extended** Message and initialize and run the **SenderResponseTimer**.

On exit from the **PE_SRC_Get_Sink_Cap_Ext** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (capabilities or response timeout).

The Policy Engine **Shall** transition back to the **PE_SRC_Ready** state (see [Figure 8-134 “Source Port State Diagram”](#)) when:

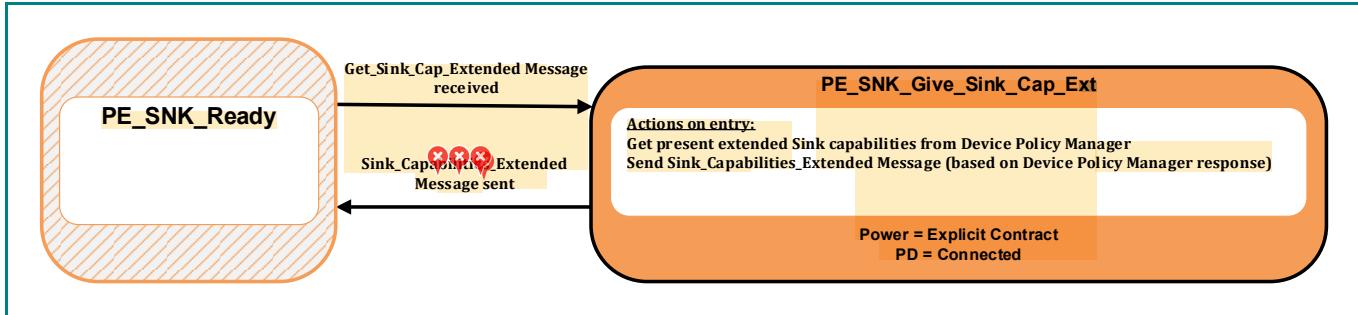
- A **Sink_Capabilities_Extended** Message is received
- Or **SenderResponseTimer** times out.

8.3.3.9.4

Sink Give Sink Capabilities Extended State Diagram

Figure 8-150 “Sink Give Sink Capabilities Extended State Diagram” shows the state diagram for a Source on receiving a **Get_Sink_Cap_Extended** Message. See also *Section 6.5.13 “Sink_Capabilities_Extended Message”*.

Figure 8-150 “Sink Give Sink Capabilities Extended State Diagram”



8.3.3.9.4.1

PE_SNK_Give_Sink_Cap_Ext State

The Policy Engine **Shall** transition to the **PE_SNK_Give_Sink_Cap_Ext** state, from the **PE_SNK_Ready** state, when a **Get_Sink_Cap_Extended** Message is received.

On entry to the **PE_SNK_Give_Sink_Cap_Ext** state the Policy Engine **Shall** request the present extended Source capabilities from the Device Policy Manager and then send a **Sink_Capabilities_Extended** Message based on these capabilities.

The Policy Engine **Shall** transition back to the **PE_SNK_Ready** state (see *Figure 8-135 “Sink Port State Diagram”*) when:

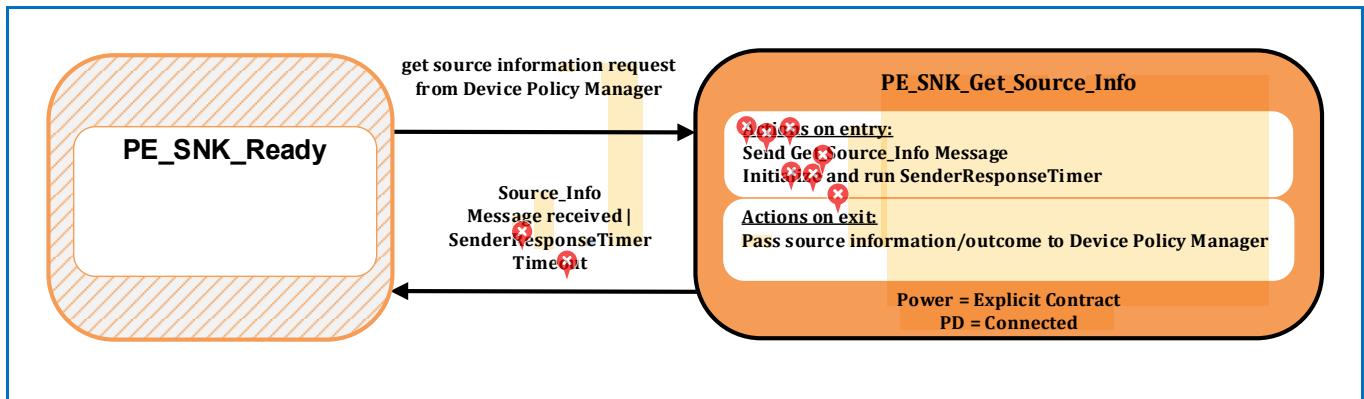
- The **Sink_Capabilities_Extended** Message has been successfully sent.

8.3.3.10 Source Information State Diagrams

8.3.3.10.1 Sink Port Get Source Information State Diagram

Figure 8-151 “Sink Port Get Source Information State Diagram” shows the state diagram for a Sink on receiving a request from the Device Policy Manager to get the Port Partner’s Source information. See also [Section 6.3.23 “Get_Source_Info Message”](#) and [Section 6.4.11 “Source_Info Message”](#).

Figure 8-151 “Sink Port Get Source Information State Diagram”



8.3.3.10.1.1 PE_SNK_Get_Source_Info State

The Policy Engine **Shall** transition to the **PE_SNK_Get_Source_Info** state, from the **PE_SNK_Ready** state, due to a request to get the remote source information from the Device Policy Manager.

On entry to the **PE_SNK_Get_Source_Info** state the Policy Engine **Shall** send a **Get_Source_Info** Message and initialize and run the **SenderResponseTimer**.

On exit from the **PE_SNK_Get_Source_Info** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (information or response timeout).

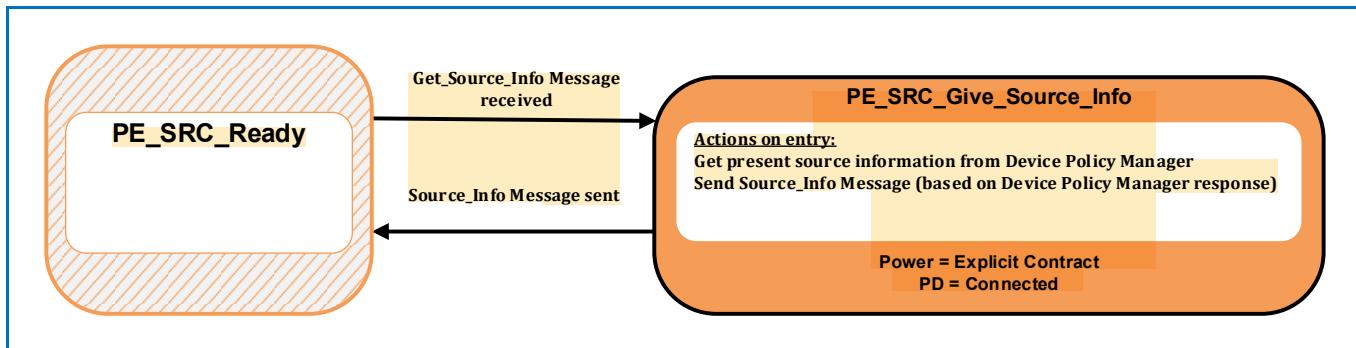
The Policy Engine **Shall** transition back to the **PE_SNK_Ready** state (see [Figure 8-135 “Sink Port State Diagram”](#)) when:

- A **Source_Info** Message is received
- Or **SenderResponseTimer** times out.

8.3.3.10.2 | Source Give Source Information State Diagram

Figure 8-152 “Source Give Source Information State Diagram” shows the state diagram for a Source on receiving a *Get_Source_Info* Message. See also [Section 6.3.23 “Get_Source_Info Message”](#) and [Section 6.4.11 “Source_Info Message”](#).

Figure 8-152 “Source Give Source Information State Diagram”



8.3.3.10.2.1 PE_SRC_Give_Source_Info State

The Policy Engine **Shall** transition to the **PE_SRC_Give_Source_Info** state, from the **PE_SRC_Ready** state, when a *Get_Source_Info* Message is received.

On entry to the **PE_SRC_Give_Source_Info** state the Policy Engine **Shall** request the present Source information from the Device Policy Manager and then send a *Source_Info* Message based on this information.

The Policy Engine **Shall** transition back to the **PE_SRC_Ready** state (see [Figure 8-134 “Source Port State Diagram”](#)) when:

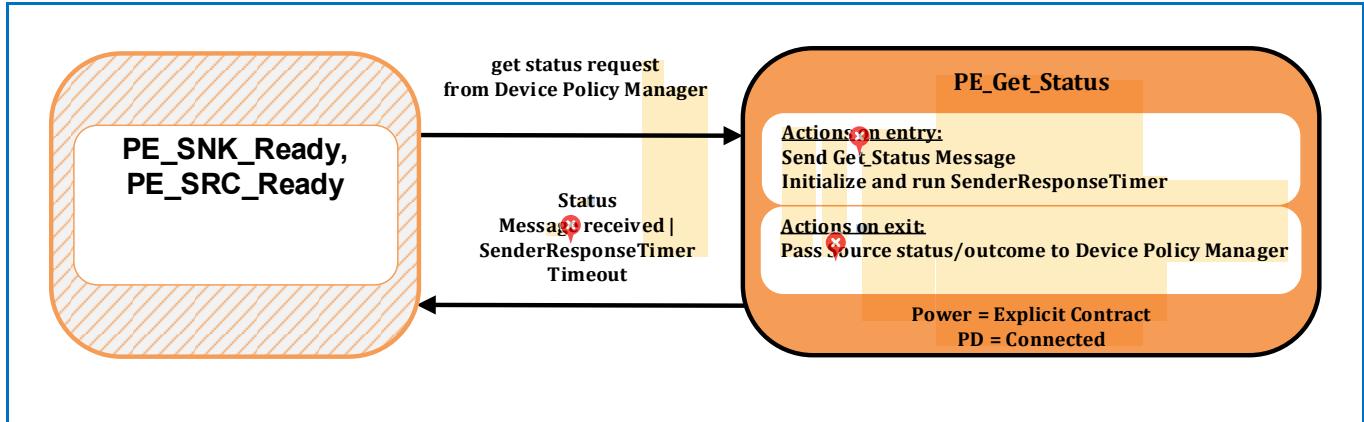
- The *Source_Info* Message has been successfully sent.

8.3.3.11 Status State Diagrams

8.3.3.11.1 Get Status State Diagram

Figure 8-153 “Get Status State Diagram” shows the state diagram for a Port on receiving a request from the Device Policy Manager to get the Port Partner or Cable Plug’s Status. See also [Section 6.5.2 “Status Message”](#).

Figure 8-153 “Get Status State Diagram”



8.3.3.11.1.1 PE_Get_Status State

The Policy Engine **Shall** transition to the **PE_Get_Status** state, from the **PE_SRC_Ready** or **PE_SNK_Ready** States, due to a request to get the Port Partner or Cable Plug’s status from the Device Policy Manager.

On entry to the **PE_Get_Status** state the Policy Engine **Shall** send a **Get_Status** Message and initialize and run the **SenderResponseTimer**.

On exit from the **PE_Get_Status** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (status or response timeout).

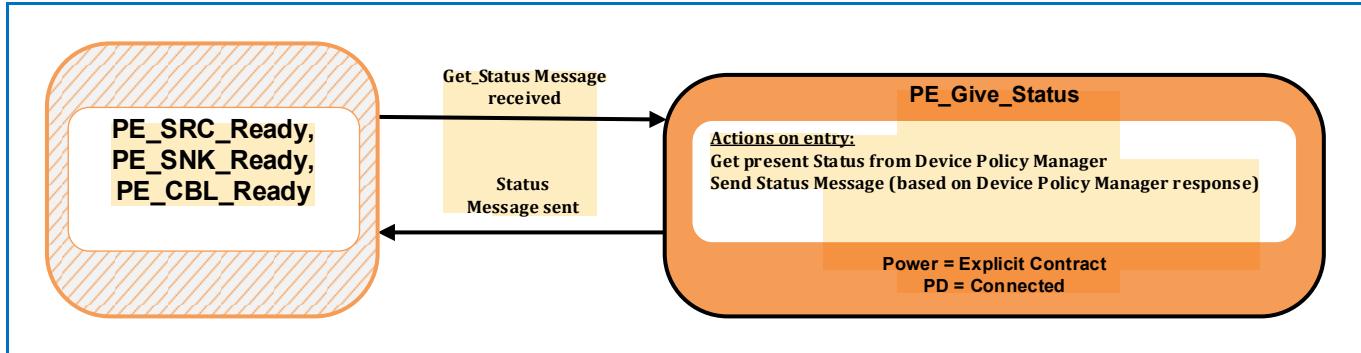
The Policy Engine **Shall** transition back to the **PE_SRC_Ready** or **PE_SNK_Ready** States as appropriate (see [Figure 8-134 “Source Port State Diagram”](#) or [Figure 8-135 “Sink Port State Diagram”](#)) when:

- A **Status** Message is received
- Or **SenderResponseTimer** times out.

8.3.3.11.2 Give Status State Diagram

[Figure 8-154 “Give Status State Diagram”](#) shows the state diagram for a Source on receiving a *Get_Status* Message. See also [Section 6.5.2 “Status Message”](#).

[Figure 8-154 “Give Status State Diagram”](#)



8.3.3.11.2.1 PE_Give_Status State

The Policy Engine *Shall* transition to the *PE_Give_Status* state, from the *PE_SRC_Ready*, *PE_SNK_Ready* or *PE_CBL_Ready* States, when a *Get_Status* Message is received.

On entry to the *PE_Give_Status* state the Policy Engine *Shall* request the present Source status from the Device Policy Manager and then send a *Status* Message based on these capabilities.

The Policy Engine *Shall* transition back to the *PE_SRC_Ready*, *PE_SNK_Ready* or *PE_CBL_Ready* States as appropriate (see [Figure 8-134 “Source Port State Diagram”](#), [Figure 8-135 “Sink Port State Diagram”](#) and [Figure 8-206 “Cable Ready State Diagram”](#)) when:

- The *Status* Message has been successfully sent.

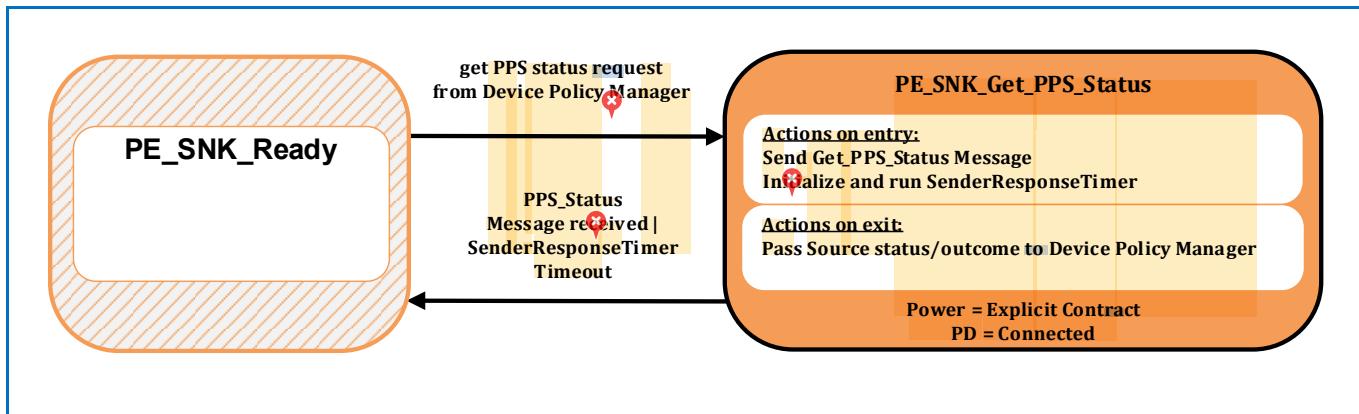


8.3.3.11.3

Sink Port Get Source PPS Status State Diagram

[Figure 8-155 “Sink Port Get Source PPS Status State Diagram”](#) shows the state diagram for a Sink on receiving a request from the Device Policy Manager to get the Port Partner’s Source status when operating as a PPS. See also [Section 6.5.10 “PPS_Status Message”](#).

[Figure 8-155 “Sink Port Get Source PPS Status State Diagram”](#)



8.3.3.11.3.1

PE_SNK_Get_PPS_Status State

The Policy Engine **Shall** transition to the **PE_SNK_Get_PPS_Status** state, from the **PE_SNK_Ready** state, due to a request to get the remote source PPS status from the Device Policy Manager.

On entry to the **PE_SNK_Get_PPS_Status** state the Policy Engine **Shall** send a **Get_PPS_Status** Message and initialize and run the **SenderResponseTimer**.

On exit from the **PE_SNK_Get_PPS_Status** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (status or response timeout).

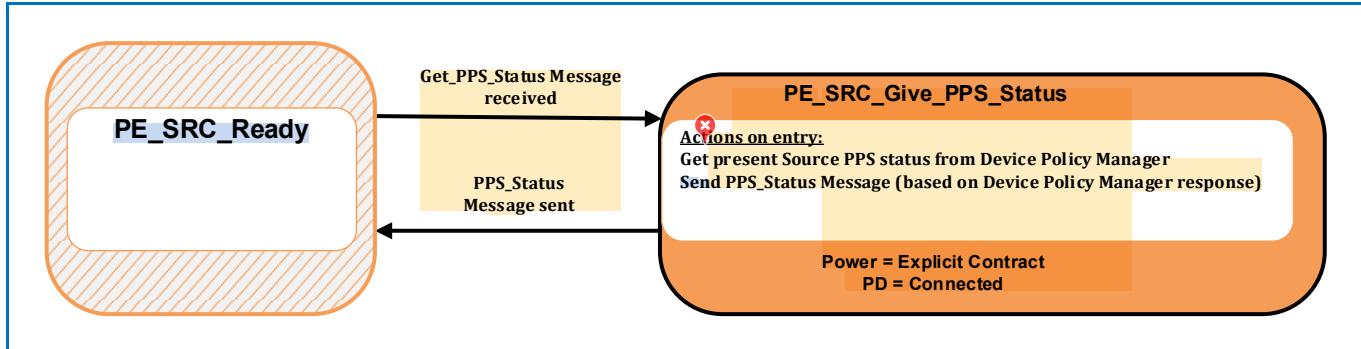
The Policy Engine **Shall** transition back to the **PE_SNK_Ready** state (see [Figure 8-135 “Sink Port State Diagram”](#)) when:

- A **PPS_Status** Message is received
- Or **SenderResponseTimer** times out.

8.3.3.11.4 Source Give Source PPS Status State Diagram

Figure 8-156 “Source Give Source PPS Status State Diagram” shows the state diagram for a Source on receiving a *Get_PPS_Status* Message. See also [Section 6.5.10 “PPS_Status Message”](#).

Figure 8-156 “Source Give Source PPS Status State Diagram”



8.3.3.11.4.1 PE_SRC_Give_PPS_Status State

The Policy Engine **Shall** transition to the **PE_SRC_Give_PPS_Status** state, from the **PE_SRC_Ready** state, when a *Get_PPS_Status* Message is received.

On entry to the **PE_SRC_Give_PPS_Status** state the Policy Engine **Shall** request the present Source PPS status from the Device Policy Manager and then send a *PPS_Status* Message based on these capabilities.

The Policy Engine **Shall** transition back to the **PE_SRC_Ready** state (see [Figure 8-134 “Source Port State Diagram”](#)) when:

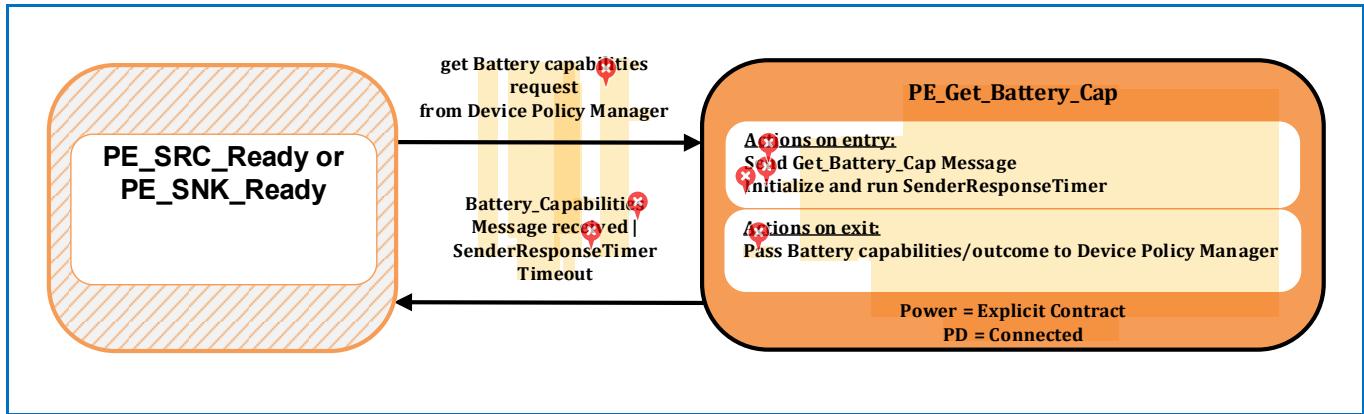
- The *PPS_Status* Message has been successfully sent.

8.3.3.12 Battery Capabilities State Diagrams

8.3.3.12.1 Get Battery Capabilities State Diagram

Figure 8-157 “Get Battery Capabilities State Diagram” shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to get the Port Partner’s Battery capabilities for a specified Battery. See also [Section 6.5.5 “Battery Capabilities Message”](#).

Figure 8-157 “Get Battery Capabilities State Diagram”



8.3.3.12.1.1 PE_Get_Battery_Cap State

The Policy Engine **Shall** transition to the **PE_Get_Battery_Cap** state, from either the **PE_SRC_Ready** or **PE_SNK_Ready** state, due to a request to get the remote Battery capabilities, for a specified Battery, from the Device Policy Manager.

On entry to the **PE_Get_Battery_Cap** state the Policy Engine **Shall** send a **Get_Battery_Cap** Message and initialize and run the **SenderResponseTimer**.

On exit from the **PE_Get_Battery_Cap** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (capabilities or response timeout).

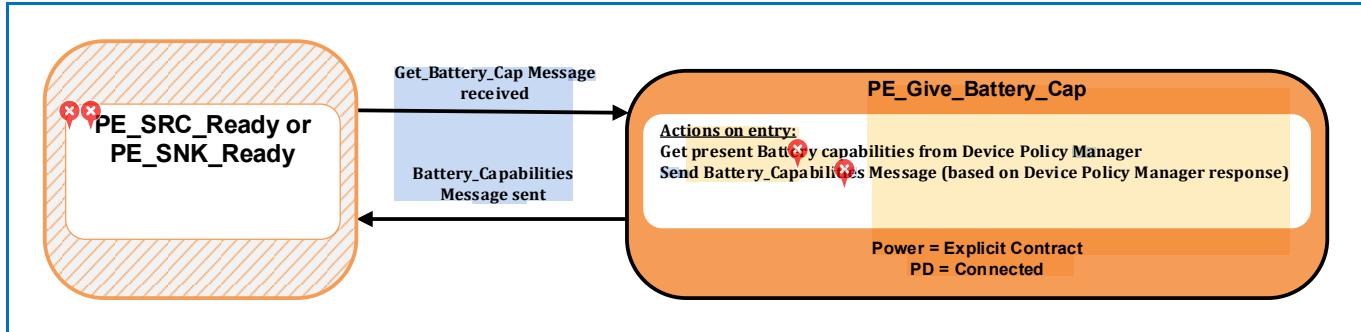
The Policy Engine **Shall** transition back to either the **PE_SRC_Ready** or **PE_SNK_Ready** state as appropriate (see [Figure 8-134 “Source Port State Diagram”](#) and [Figure 8-135 “Sink Port State Diagram”](#)) when:

- A **Battery_Capabilities** Message is received
- Or **SenderResponseTimer** times out.

8.3.3.12.2 Give Battery Capabilities State Diagram

Figure 8-158 “Give Battery Capabilities State Diagram” shows the state diagram for a Source or Sink on receiving a **Get_Battery_Cap** Message. See also Section 6.5.5 “Battery_Capabilities Message”.

Figure 8-158 “Give Battery Capabilities State Diagram”



8.3.3.12.2.1 PE_Give_Battery_Cap State

The Policy Engine **Shall** transition to the **PE_Give_Battery_Cap** state, from either the **PE_SRC_Ready** or **PE_SNK_Ready** state, when a **Get_Battery_Cap** Message is received.

On entry to the **PE_Give_Battery_Cap** state the Policy Engine **Shall** request the present Battery capabilities, for the requested Battery, from the Device Policy Manager and then send a **Battery_Capabilities** Message based on these capabilities.

The Policy Engine **Shall** transition back to either the **PE_SRC_Ready** or **PE_SNK_Ready** state as appropriate (see Figure 8-134 “Source Port State Diagram” and Figure 8-135 “Sink Port State Diagram”) when:

- The **Battery_Capabilities** Message has been successfully sent.

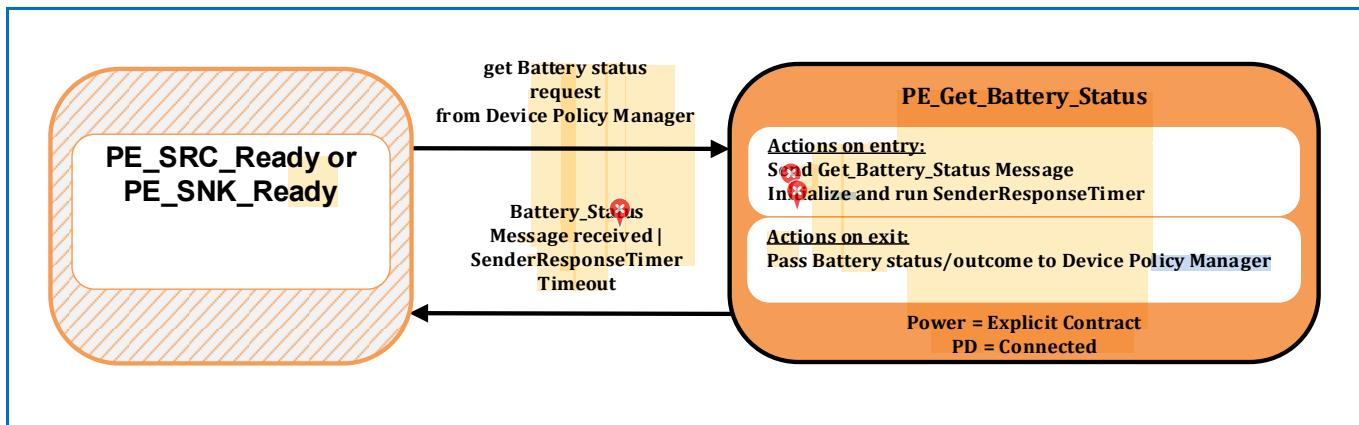
8.3.3.13

Battery Status State Diagrams

8.3.3.13.1 Get Battery Status State Diagram

Figure 8-159 “Get Battery Status State Diagram” shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to get the Port Partner’s Battery status for a specified Battery. See also Section 6.5.4 “Get_Battery_Status Message”.

Figure 8-159 “Get Battery Status State Diagram”



8.3.3.13.1.1 PE_Get_Battery_Status State

The Policy Engine **Shall** transition to the **PE_Get_Battery_Status** state, from either the **PE_SRC_Ready** or **PE_SNK_Ready** state, due to a request to get the remote Battery status, for a specified Battery, from the Device Policy Manager.

On entry to the **PE_Get_Battery_Status** state the Policy Engine **Shall** send a **Get_Battery_Status** Message and initialize and run the **SenderResponseTimer**.

On exit from the **PE_Get_Battery_Status** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (status or response timeout).

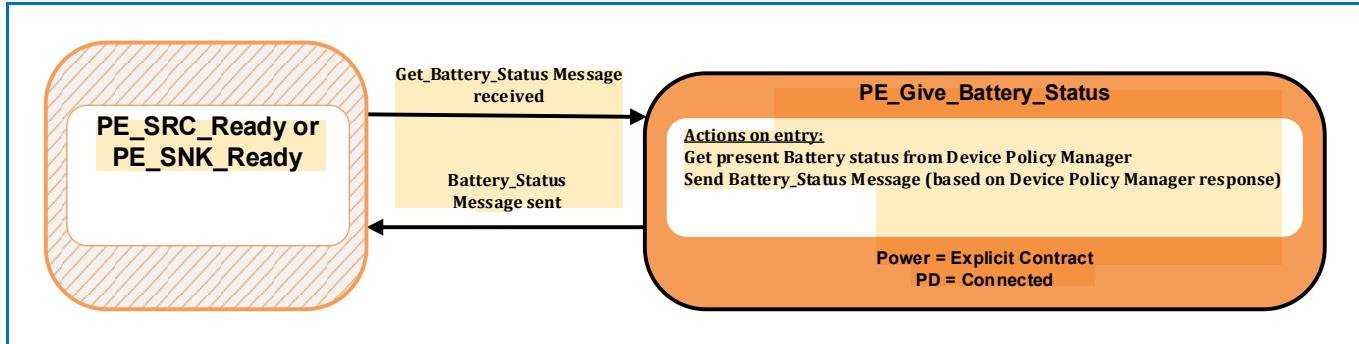
The Policy Engine **Shall** transition back to either the **PE_SRC_Ready** or **PE_SNK_Ready** state as appropriate (see Figure 8-134 “Source Port State Diagram” and Figure 8-135 “Sink Port State Diagram”) when:

- A **Battery_Status** Message is received
- Or **SenderResponseTimer** times out.

8.3.3.13.2 Give Battery Status State Diagram

Figure 8-160 “Give Battery Status State Diagram” shows the state diagram for a Source or Sink on receiving a *Get_Battery_Status* Message. See also [Section 6.5.4 “Get_Battery_Status Message”](#).

Figure 8-160 “Give Battery Status State Diagram”



8.3.3.13.2.1 PE_Give_Battery_Status State

The Policy Engine **Shall** transition to the **PE_Give_Battery_Status** state, from either the **PE_SRC_Ready** or **PE_SNK_Ready** state, when a **Battery_Status** Message is received.

On entry to the **PE_Give_Battery_Status** state the Policy Engine **Shall** request the present Battery status, for the requested Battery, from the Device Policy Manager and then send a **Battery_Status** Message based on this status.

✖ The Policy Engine **Shall** transition back to either the **PE_SRC_Ready** or **PE_SNK_Ready** state as appropriate (see [Figure 8-134 “Source Port State Diagram”](#) and [Figure 8-135 “Sink Port State Diagram”](#)) when:

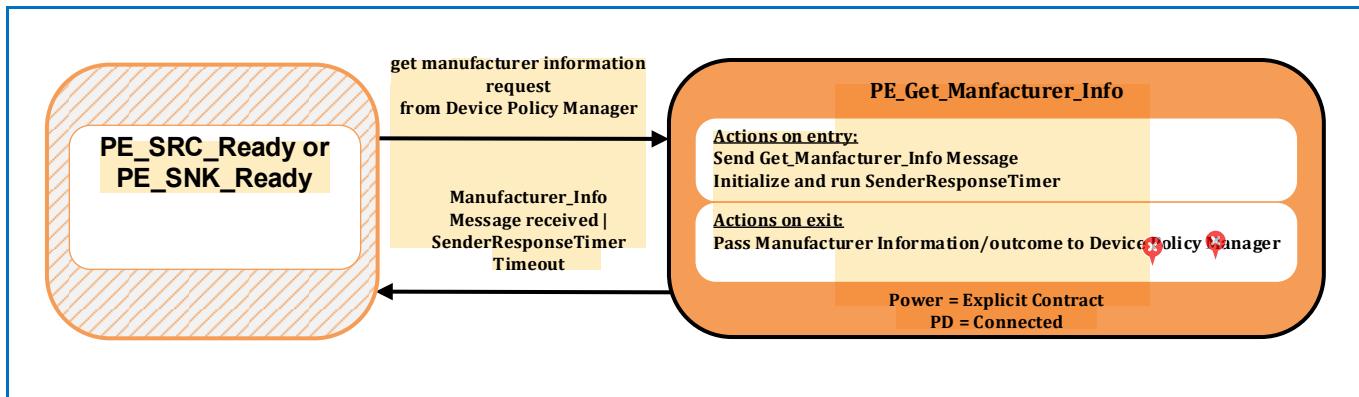
- The **Battery_Status** Message has been successfully sent.

8.3.3.14 Manufacturer Information State Diagrams

8.3.3.14.1 Get Manufacturer Information State Diagram

Figure 8-161 “Get Manufacturer Information State Diagram” shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to get the Port Partner or Cable Plug’s Manufacturer Information. See also [Section 6.5.6 “Get_Manufacturer_Info Message”](#).

Figure 8-161 “Get Manufacturer Information State Diagram”



8.3.3.14.1.1 PE_Get_Manufacturer_Info State

The Policy Engine **Shall** transition to the **PE_Get_Manufacturer_Info** state, from either the **PE_SRC_Ready** or **PE_SNK_Ready** state, due to a request to get the remote Manufacturer Information from the Device Policy Manager.

On entry to the **PE_Get_Manufacturer_Info** state the Policy Engine **Shall** send a **Get_Manufacturer_Info** Message and initialize and run the **SenderResponseTimer**.

On exit from the **PE_Get_Manufacturer_Info** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (information or response timeout).

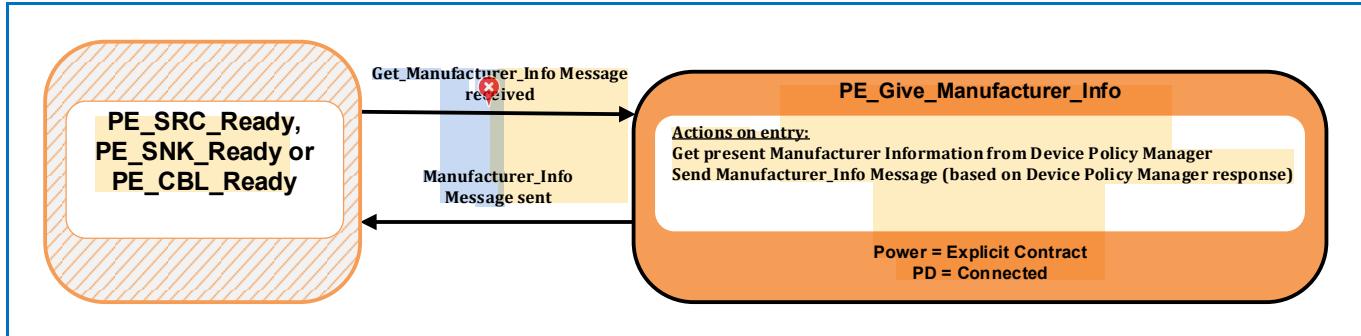
The Policy Engine **Shall** transition back to either the **PE_SRC_Ready** or **PE_SNK_Ready** state as appropriate (see [Figure 8-134 “Source Port State Diagram”](#) and [Figure 8-135 “Sink Port State Diagram”](#)) when:

- A **Manufacturer_Info** Message is received
- Or **SenderResponseTimer** times out.

8.3.3.14.2 Give Manufacturer Information State Diagram

Figure 8-162 “Give Manufacturer Information State Diagram” shows the state diagram for a Source, Sink or Cable Plug on receiving a **Get_Manufacturer_Info** Message. See also [Section 6.5.6 “Get_Manufacturer_Info Message”](#).

Figure 8-162 “Give Manufacturer Information State Diagram”



8.3.3.14.2.1 PE_Give_Manufacturer_Info State

• The Policy Engine **Shall** transition to the **PE_Give_Manufacturer_Info** state, from either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state, when a **Get_Manufacturer_Info** Message is received.

On entry to the **PE_Give_Manufacturer_Info** state the Policy Engine **Shall** request the manufacturer information from the Device Policy Manager and then send a **Manufacturer_Info** Message based on this status.

The Policy Engine **Shall** transition back to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state as appropriate (see [Figure 8-134 “Source Port State Diagram”](#), [Figure 8-135 “Sink Port State Diagram”](#) and [Figure 8-206 “Cable Ready State Diagram”](#)) when:

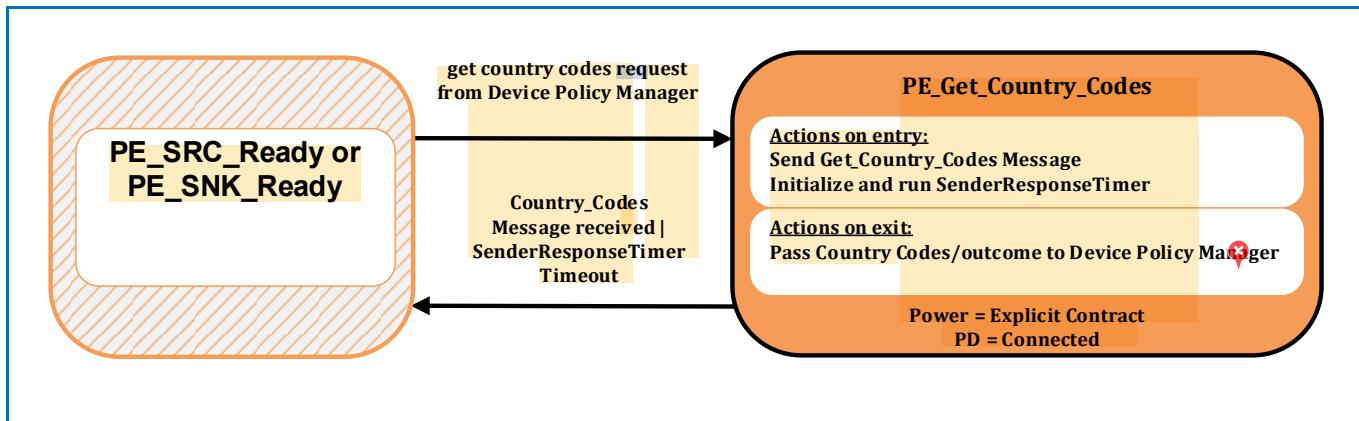
- The **Manufacturer_Info** Message has been successfully sent.

8.3.3.15 Country Codes and Information State Diagrams

8.3.3.15.1 Get Country Codes State Diagram

Figure 8-163 “Get Country Codes State Diagram” shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to get the Port Partner or Cable Plug’s Country Codes. See also [Section 6.5.11 “Country_Codes Message”](#).

Figure 8-163 “Get Country Codes State Diagram”



8.3.3.15.1.1 PE_Get_Country_Codes State

The Policy Engine **Shall** transition to the **PE_Get_Country_Codes** state, from either the **PE_SRC_Ready** or **PE_SNK_Ready** state, due to a request to get the remote Country Codes from the Device Policy Manager.

On entry to the **PE_Get_Country_Codes** state the Policy Engine **Shall** send a *Get_Country_Codes* Message and initialize and run the *SenderResponseTimer*.

On exit from the **PE_Get_Country_Codes** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (Country Codes or response timeout).

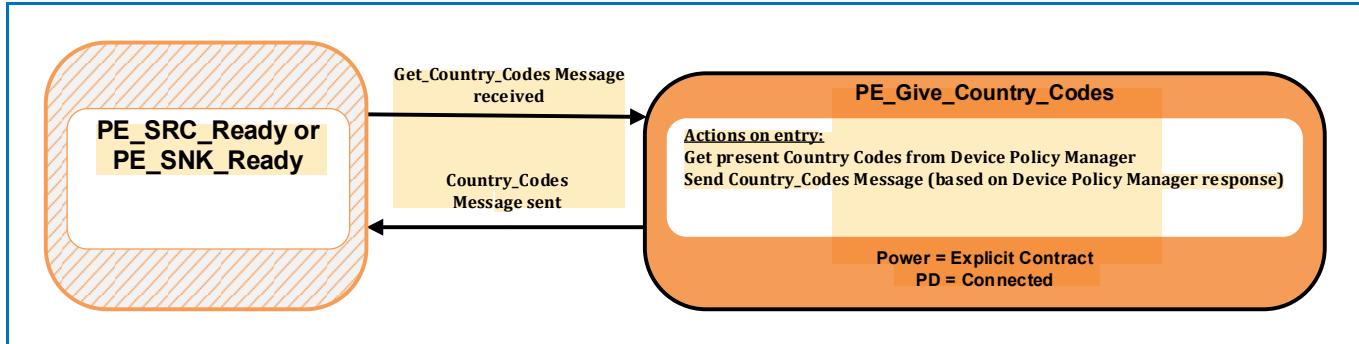
The Policy Engine **Shall** transition back to either the **PE_SRC_Ready** or **PE_SNK_Ready** state as appropriate (see [Figure 8-134 “Source Port State Diagram”](#) and [Figure 8-135 “Sink Port State Diagram”](#)) when:

- A *Country_Codes* Message is received
- Or *SenderResponseTimer* times out.

8.3.3.15.2 | Give Country Codes State Diagram

Figure 8-164 “Give Country Codes State Diagram” shows the state diagram for a Source or Sink on receiving a `Get_Country_Codes` Message. See also Section 6.5.11 “Country_Codes Message”.

Figure 8-164 “Give Country Codes State Diagram”



8.3.3.15.2.1 | PE_Give_Country_Codes State

The Policy Engine **Shall** transition to the `PE_Give_Country_Codes` state, from either the `PE_SRC_Ready` or `PE_SNK_Ready` State, when a `Get_Country_Codes` Message is received.

On entry to the `PE_Give_Country_Codes` state the Policy Engine **Shall** request the country codes from the Device Policy Manager and then send a `Country_Codes` Message containing these codes.

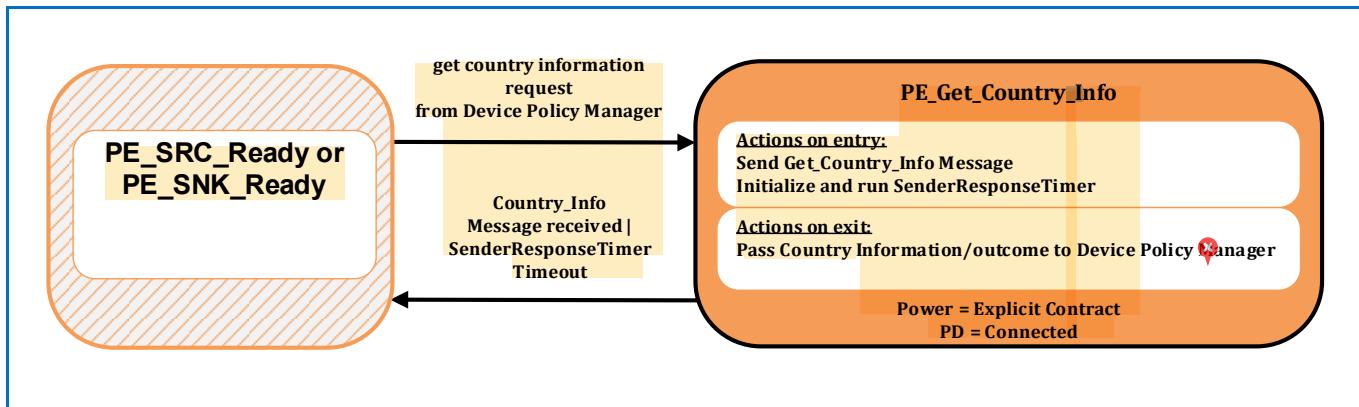
The Policy Engine **Shall** transition back to either the `PE_SRC_Ready` or `PE_SNK_Ready` State as appropriate (see Figure 8-134 “Source Port State Diagram” and Figure 8-135 “Sink Port State Diagram”) when:

- The `Country_Codes` Message has been successfully sent.

8.3.3.15.3 Get Country Information State Diagram

Figure 8-165 “Get Country Information State Diagram” shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to get the Port Partner or Cable Plug’s Country Information. See also [Section 6.5.12 “Country_Info Message”](#).

Figure 8-165 “Get Country Information State Diagram”



8.3.3.15.3.1 PE_Get_Country_Info State

The Policy Engine **Shall** transition to the **PE_Get_Country_Info** state, from either the **PE_SRC_Ready** or **PE_SNK_Ready** state, due to a request to get the remote Manufacturer Information from the Device Policy Manager.

On entry to the **PE_Get_Country_Info** state the Policy Engine **Shall** send a **Get_Manufacturer_Info** Message and initialize and run the **SenderResponseTimer**.

On exit from the **PE_Get_Country_Info** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (country information or response timeout).

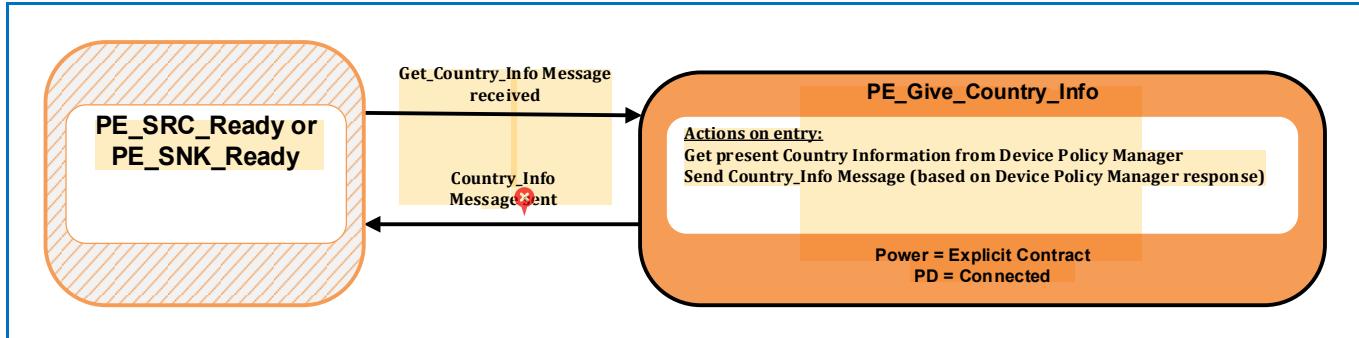
The Policy Engine **Shall** transition back to either the **PE_SRC_Ready** or **PE_SNK_Ready** state as appropriate (see [Figure 8-134 “Source Port State Diagram”](#) and [Figure 8-135 “Sink Port State Diagram”](#)) when:

- A **Country_Info** Message is received
- Or **SenderResponseTimer** times out.

8.3.3.15.4 Give Country Information State Diagram

Figure 8-166 “Give Country Information State Diagram” shows the state diagram for a Source or Sink on receiving a *Get_Country_Info* Message. See also Section 6.5.12 “Country_Info Message”.

Figure 8-166 “Give Country Information State Diagram”



8.3.3.15.4.1 PE_Give_Country_Info State

📍 The Policy Engine *Shall* transition to the *PE_Give_Country_Info* state, from either the *PE_SRC_Ready* or *PE_SNK_Ready* State, when a *Get_Country_Info* Message is received.

On entry to the *PE_Give_Country_Info* state the Policy Engine *Shall* request the country information from the Device Policy Manager and then send a *Country_Info* Message containing this country information.

The Policy Engine *Shall* transition back to either the *PE_SRC_Ready* or *PE_SNK_Ready* State as appropriate (see Figure 8-134 “Source Port State Diagram” and Figure 8-135 “Sink Port State Diagram”) when:

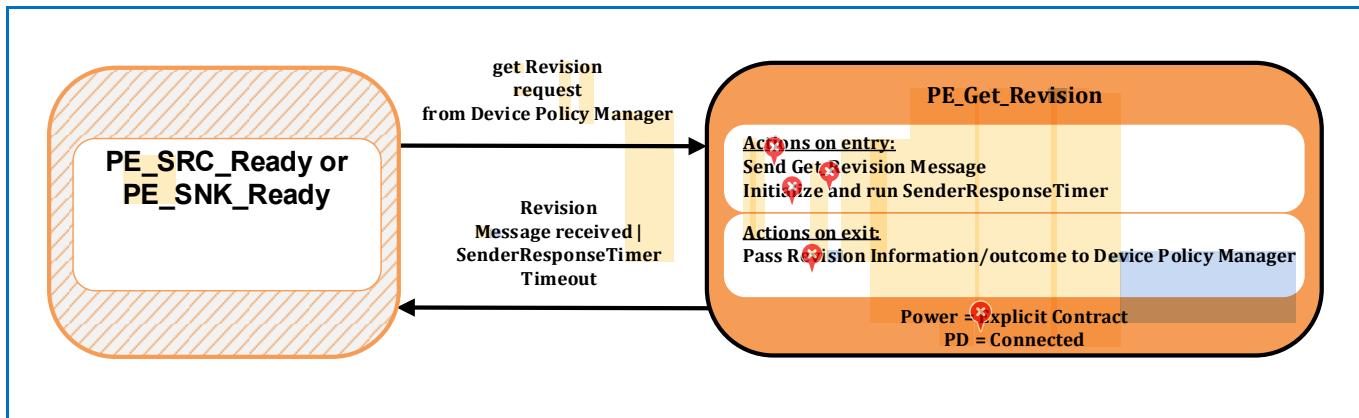
- The *Country_Info* Message has been successfully sent.

8.3.3.16 Revision State Diagrams

8.3.3.16.1 Get Revision State Diagram

Figure 8-167 “Get Revision State Diagram” shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to get the Port Partner or Cable Plug’s Revision Information. See also [Section 6.3.24 “Get_Revision Message”](#) and [Section 6.4.12 “Revision Message”](#).

Figure 8-167 “Get Revision State Diagram”



8.3.3.16.1.1 PE_Get_Revision State

The Policy Engine **Shall** transition to the **PE_Get_Revision** state, from either the **PE_SRC_Ready** or **PE_SNK_Ready** state, due to a request to get the remote Revision Information from the Device Policy Manager.

On entry to the **PE_Get_Revision** state the Policy Engine **Shall** send a **Get_Revision** Message and initialize and run the **SenderResponseTimer**.

On exit from the **PE_Get_Revision** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (Revision information or response timeout).

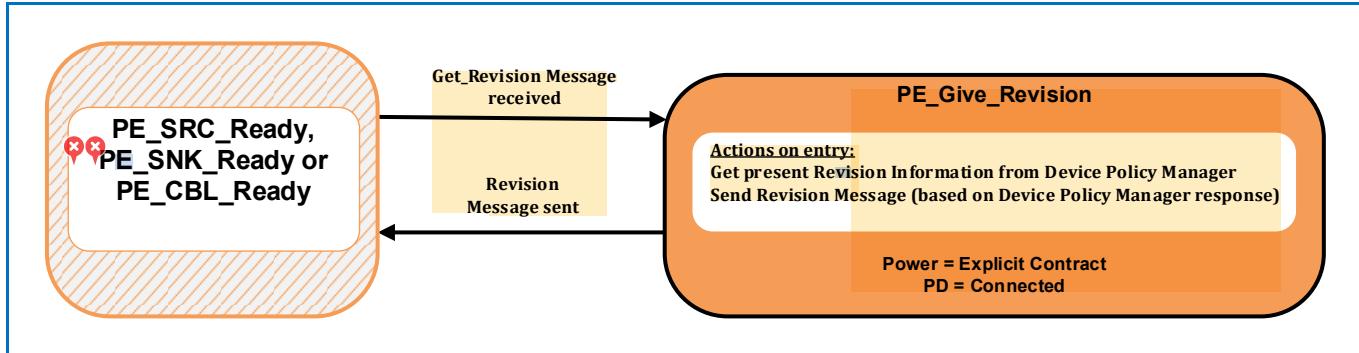
The Policy Engine **Shall** transition back to either the **PE_SRC_Ready** or **PE_SNK_Ready** state as appropriate (see [Figure 8-134 “Source Port State Diagram”](#) and [Figure 8-135 “Sink Port State Diagram”](#)) when:

- A **Revision** Message is received
- Or **SenderResponseTimer** times out.

8.3.3.16.2 Give Revision State Diagram

Figure 8-168 “Give Revision State Diagram” shows the state diagram for a Source, Sink or Cable Plug on receiving a **Get_Revision** Message. See also [Section 6.3.24 “Get_Revision Message”](#) and [Section 6.4.12 “Revision Message”](#).

Figure 8-168 “Give Revision State Diagram”



8.3.3.16.2.1 PE_Give_Revision State

The Policy Engine **Shall** transition to the **PE_Give_Revision** state, from either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state, when a **Get_Revision** Message is received.

On entry to the **PE_Give_Revision** state the Policy Engine **Shall** request the Revision information from the Device Policy Manager and then send a **Revision** Message based on this information.

The Policy Engine **Shall** transition back to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state as appropriate (see [Figure 8-134 “Source Port State Diagram”](#), [Figure 8-135 “Sink Port State Diagram”](#) and [Figure 8-206 “Cable Ready State Diagram”](#)) when:

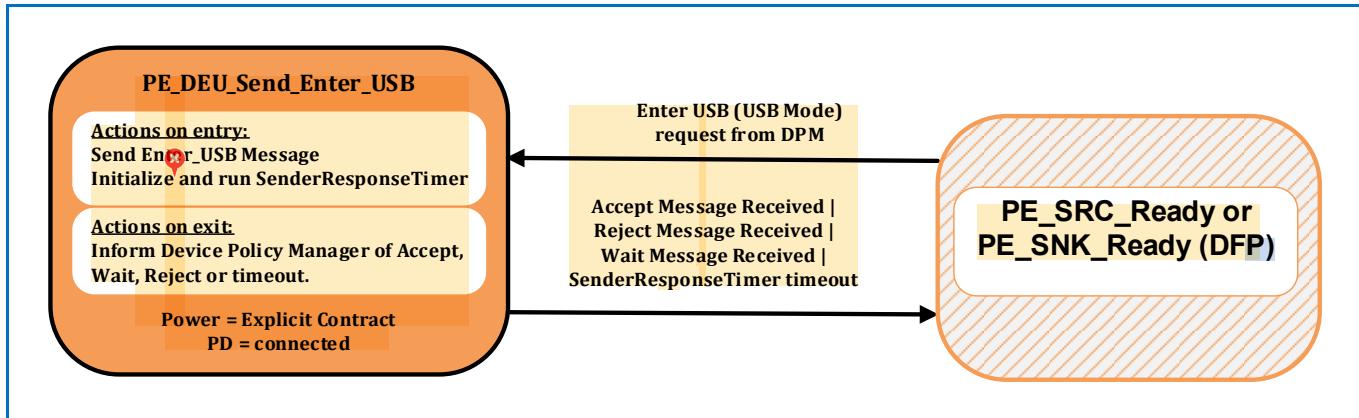
- The **Revision** Message has been successfully sent.

8.3.3.17 Enter_USB Message State Diagrams

8.3.3.17.1 DFP Enter_USB Message State Diagrams

Figure 8-169 “DFP Enter_USB Message State Diagram” shows the state diagram for an *Enter_USB* Message sent by a DFP.

Figure 8-169 “DFP Enter_USB Message State Diagram”



8.3.3.17.1.1 PE_DEU_Send_Enter_USB State

The **PE_DEU_Send_Enter_USB** State *Shall* be entered from the **PE_SRC_Ready** or **PE_SNK_Ready** State when requested by the Device Policy Manager and the Port is operating as a DFP.

On entry to the **PE_DEU_Send_Enter_USB** State the Policy Engine *Shall* request the Protocol Layer to send an *Enter_USB* Message and then initialize and run the *SenderResponseTimer*.

On exit from the **PE_DEU_Send_Enter_USB** state the Policy Engine *Shall* inform the Device Policy Manager of the outcome: *Accept* Message received, *Reject* Message received, *SenderResponseTimer* timeout.

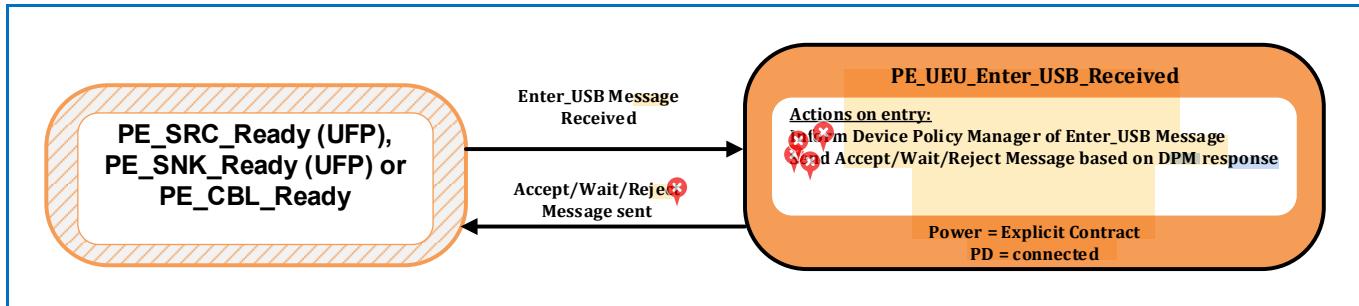
The Policy Engine *Shall* transition back to the **PE_SRC_Ready** or **PE_SNK_Ready** State depending on the Ports power role when:

- An *Accept* Message has been received or
- A *Wait* Message has been received or
- A *Reject* Message has been received
- There is a *SenderResponseTimer* timeout.

8.3.3.17.2 UFP or Cable Plug Enter_USB Message State Diagrams

Figure 8-170 “UFP Enter_USB Message State Diagram” shows the state diagram for an **Enter_USB** Message received by a UFP or Cable Plug.

Figure 8-170 “UFP Enter_USB Message State Diagram”



8.3.3.17.2.1 PE_UEU_Enter_USB_Received State

The **PE_UEU_Enter_USB_Received** state **Shall** be entered from the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state as appropriate (see [Figure 8-134 “Source Port State Diagram”](#), [Figure 8-135 “Sink Port State Diagram”](#) and [Figure 8-206 “Cable Ready State Diagram”](#)) when an **Enter_USB** Message is received and the Port is operating as a UFP or is a Cable Plug.

On entry to the **PE_UEU_Enter_USB_Received** state the Policy Engine **Shall** inform the Device Policy Manager. The Device Policy Manager responds with an indication of whether the **Enter_USB** Message is to be accepted or rejected. The Policy Engine **Shall** send either an **Accept** Message, a **Wait** Message or a **Reject** Message as appropriate.

The Policy Engine **Shall** transition back to the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state as appropriate when:

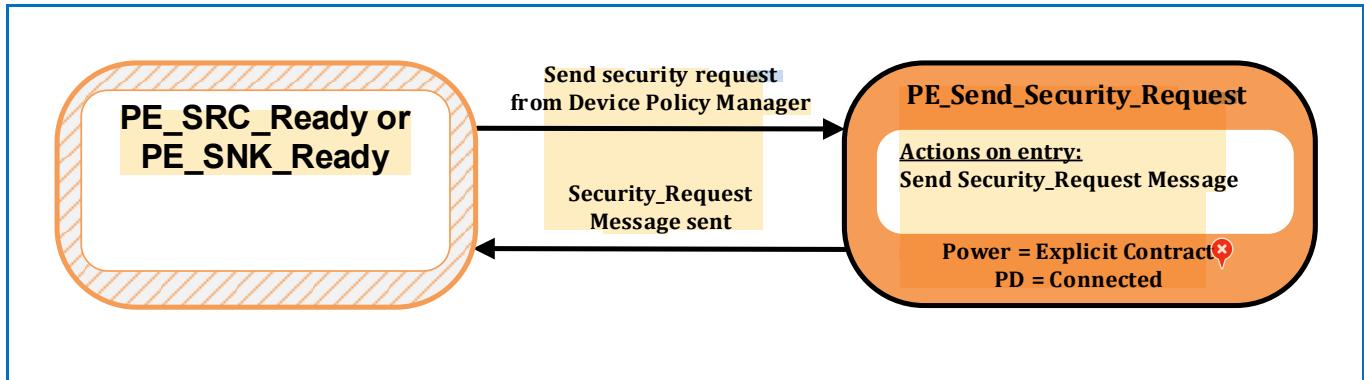
- Either an **Accept** Message, a **Wait** Message or a **Reject** Message has been sent.

8.3.3.18 Security State Diagrams

8.3.3.18.1 Send Security Request State Diagram

Figure 8-171 "Send security request State Diagram" shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to send a security request. See also [Section 6.5.8 "Security Messages"](#).

Figure 8-171 "Send security request State Diagram"



8.3.3.18.1.1 PE_Send_Security_Request State

The Policy Engine **Shall** transition to the **PE_Send_Security_Request** state, from either the **PE_SRC_Ready** or **PE_SNK_Ready** state, due to a request to send a security request from the Device Policy Manager.

On entry to the **PE_Send_Security_Request** state the Policy Engine **Shall** send a **Security_Request** Message.

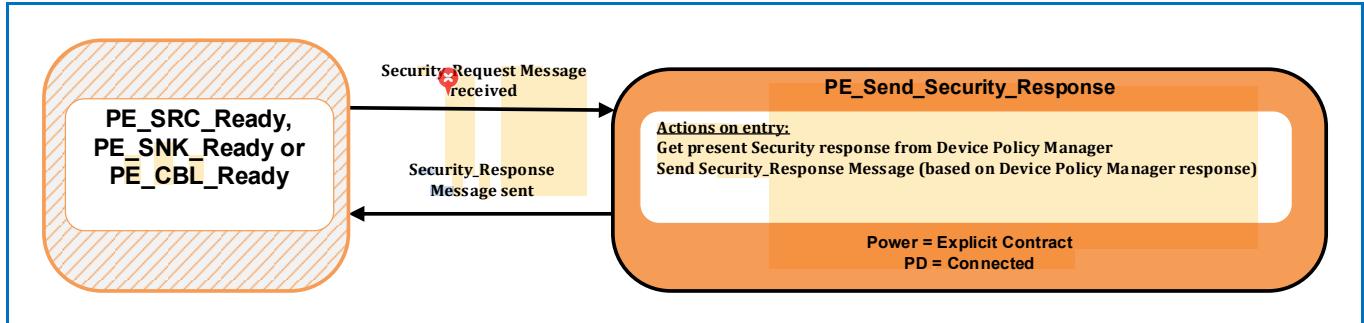
The Policy Engine **Shall** transition back to either the **PE_SRC_Ready** or **PE_SNK_Ready** state as appropriate (see [Figure 8-134 "Source Port State Diagram"](#) and [Figure 8-135 "Sink Port State Diagram"](#)) when:

- The **Security_Request** Message has been sent.

8.3.3.18.2 Send Security Response State Diagram

Figure 8-172 “Send security response State Diagram” shows the state diagram for a Source, Sink or Cable Plug on receiving a **Security_Request** Message. See also *Section 6.5.8 “Security Messages”*.

Figure 8-172 “Send security response State Diagram”



8.3.3.18.2.1 PE_Send_Security_Response State

The Policy Engine **Shall** transition to the **PE_Send_Security_Response** state, from either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state, when a **Security_Request** Message is received.

On entry to the **PE_Send_Security_Response** state the Policy Engine **Shall** request the appropriate response from the Device Policy Manager and then send a **Security_Response** Message based on this status.

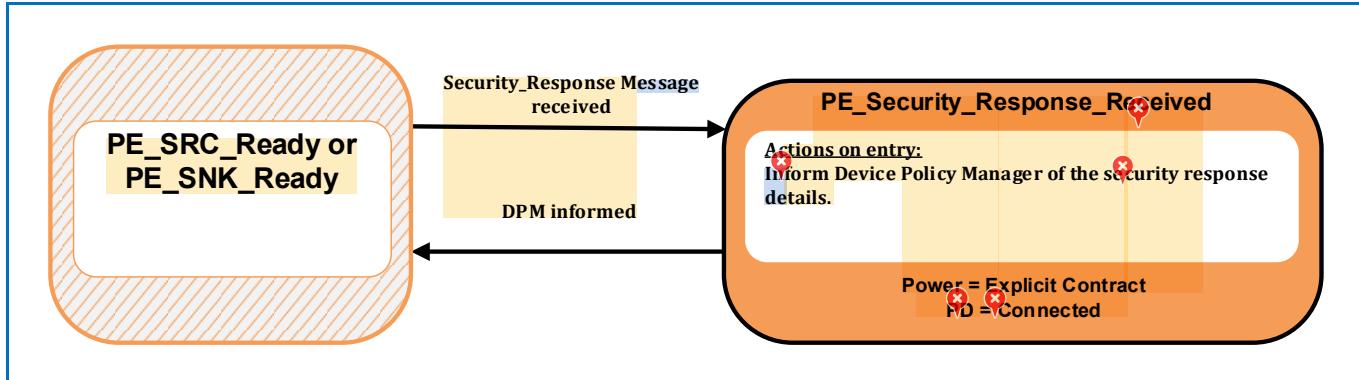
The Policy Engine **Shall** transition back to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state as appropriate (see *Figure 8-134 “Source Port State Diagram”*, *Figure 8-135 “Sink Port State Diagram”* and *Figure 8-206 “Cable Ready State Diagram”*) when:

- The **Security_Response** Message has been successfully sent.

8.3.3.18.3 Security Response Received State Diagram

Figure 8-173 “Security response received State Diagram” shows the state diagram for a Source or Sink on receiving a **Security_Response** Message. See also *Section 6.5.8 “Security Messages”*.

Figure 8-173 “Security response received State Diagram”



8.3.3.18.3.1 PE_Security_Response_Received State

The Policy Engine **Shall** transition to the **PE_Security_Response_Received** state, from either the **PE_SRC_Ready** or **PE_SNK_Ready** when a **Security_Response** Message is received.

On entry to the **PE_Security_Response_Received** state the Policy Engine **Shall** inform the Device Policy Manager of the details of the security response.

The Policy Engine **Shall** transition back to either the **PE_SRC_Ready** or **PE_SNK_Ready** state as appropriate (see *Figure 8-134 “Source Port State Diagram”*, *Figure 8-135 “Sink Port State Diagram”* and *Figure 8-206 “Cable Ready State Diagram”*) when:

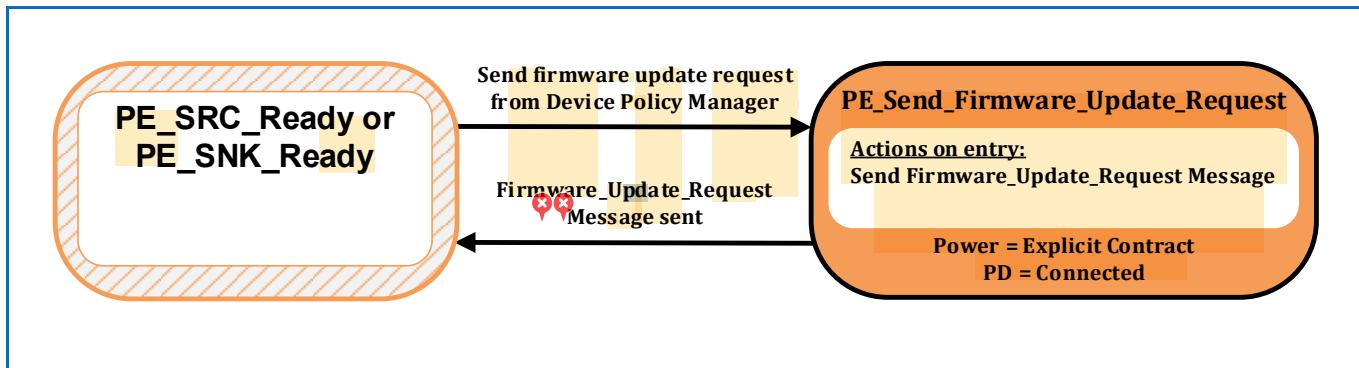
- The Device Policy Manager has been informed.

8.3.3.19 Firmware Update State Diagrams

8.3.3.19.1 Send Firmware Update Request State Diagram

Figure 8-174 “Send firmware update request State Diagram” shows the state diagram for a Source or Sink on receiving a request from the Device Policy Manager to send a firmware update request. See also [Section 6.5.9 “Firmware Update Messages”](#).

Figure 8-174 “Send firmware update request State Diagram”



8.3.3.19.1.1 PE_Send_Firmware_Update_Request State

The Policy Engine **Shall** transition to the **PE_Send_Firmware_Update_Request** state, from either the **PE_SRC_Ready** or **PE_SNK_Ready** state, due to a request to send a firmware update request from the Device Policy Manager.

On entry to the **PE_Send_Firmware_Update_Request** state the Policy Engine **Shall** send a **Firmware_Update_Request** Message.

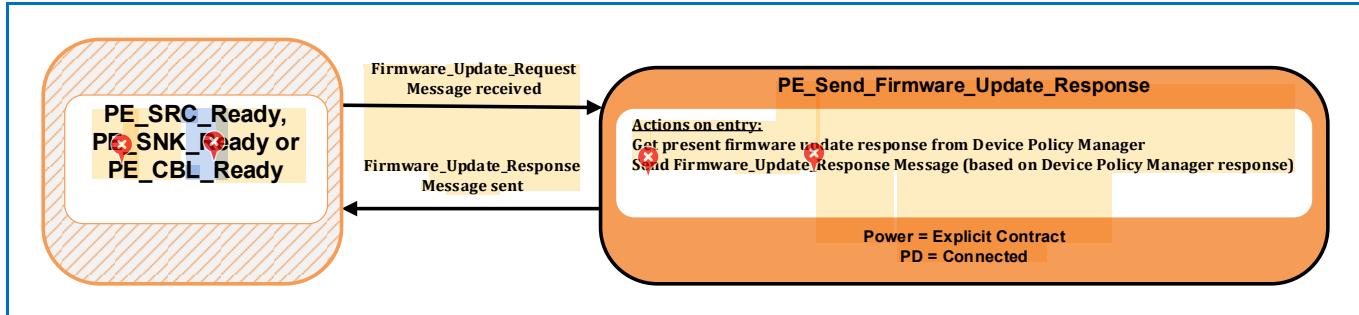
The Policy Engine **Shall** transition back to either the **PE_SRC_Ready** or **PE_SNK_Ready** state as appropriate (see [Figure 8-134 “Source Port State Diagram”](#) and [Figure 8-135 “Sink Port State Diagram”](#)) when:

- The **Firmware_Update_Request** Message has been sent.

8.3.3.19.2 | Send Firmware Update Response State Diagram

Figure 8-175 “Send firmware update response State Diagram” shows the state diagram for a Source, Sink or Cable Plug on receiving a **Firmware_Update_Request** Message. See also *Section 6.5.9 “Firmware Update Messages”*.

Figure 8-175 “Send firmware update response State Diagram”



8.3.3.19.2.1 PE_Send_Firmware_Update_Response State

The Policy Engine **Shall** transition to the **PE_Send_Firmware_Update_Response** state, from either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state, when a **Firmware_Update_Request** Message is received.

On entry to the **PE_Send_Firmware_Update_Response** state the Policy Engine **Shall** request the appropriate response from the Device Policy Manager and then send a **Firmware_Update_Response** Message based on this status.

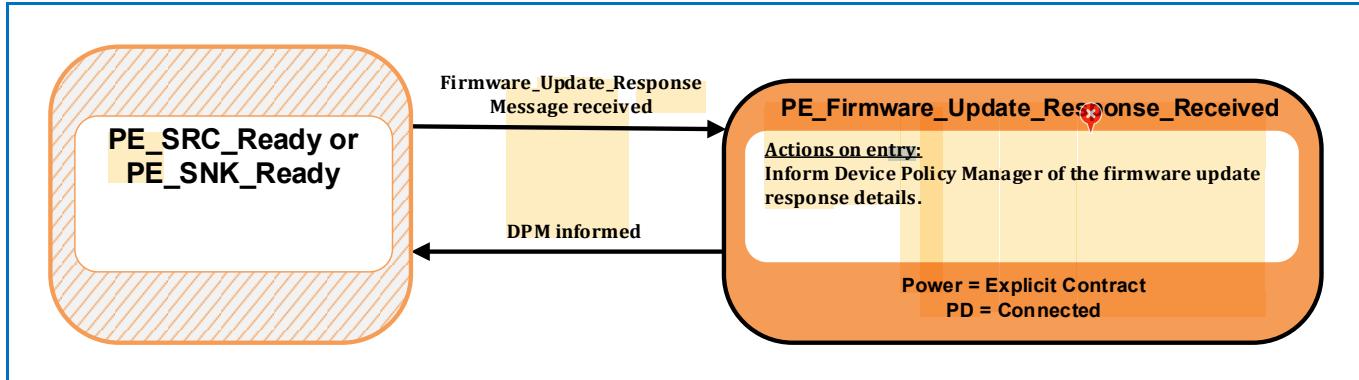
The Policy Engine **Shall** transition back to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state as appropriate (see *Figure 8-134 “Source Port State Diagram”*, *Figure 8-135 “Sink Port State Diagram”* and *Figure 8-206 “Cable Ready State Diagram”*) when:

- The **Firmware_Update_Response** Message has been successfully sent.

8.3.3.19.3 Firmware Update Response Received State Diagram

Figure 8-176 “Firmware update response received State Diagram” shows the state diagram for a Source or Sink on receiving a **Firmware_Update_Response** Message. See also *Section 6.5.9 “Firmware Update Messages”*.

Figure 8-176 “Firmware update response received State Diagram”



8.3.3.19.3.1 PE_Firmware_Update_Response_Received State

The Policy Engine **Shall** transition to the **PE_Firmware_Update_Response_Received** state, from either the **PE_SRC_Ready** or **PE_SNK_Ready** when a **Firmware_Update_Response** Message is received.

On entry to the **PE_Firmware_Update_Response_Received** state the Policy Engine **Shall** inform the Device Policy Manager of the details of the firmware update response.

The Policy Engine **Shall** transition back to either the **PE_SRC_Ready** or **PE_SNK_Ready** state as appropriate (see *Figure 8-134 “Source Port State Diagram”*, *Figure 8-135 “Sink Port State Diagram”* and *Figure 8-206 “Cable Ready State Diagram”*) when:

- The Device Policy Manager has been informed.

8.3.3.20 Dual-Role Port State Diagrams

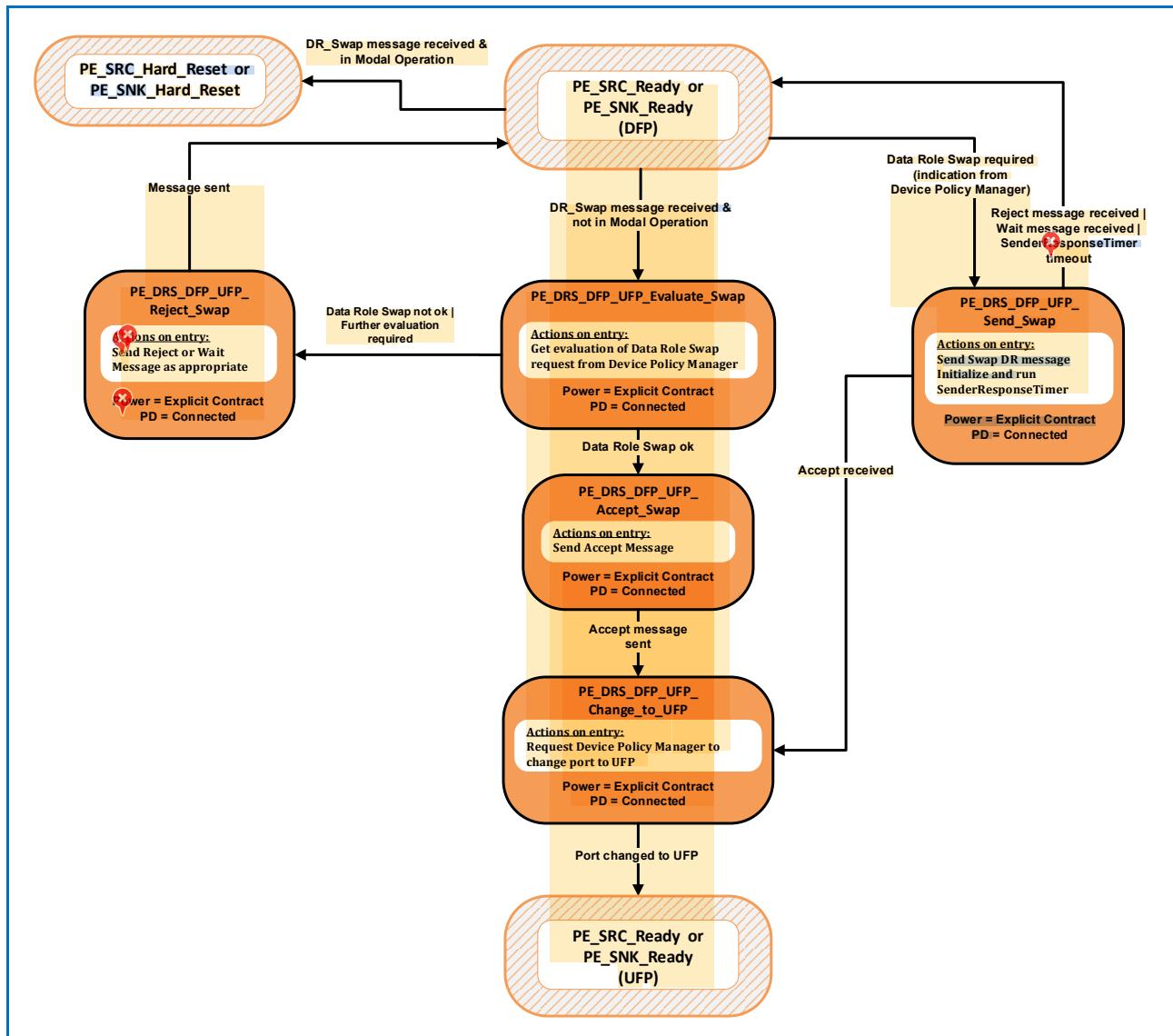
Dual-Role Ports that combine Source and Sink capabilities **Shall** comprise Source and Sink Policy Engine state machines. In addition they **Shall** have the capability to perform a Power Role Swap from the **PE_SRC_Ready** or **PE_SNK_Ready** states and **Shall** return to USB Default Operation on a Hard Reset.

The State Diagrams in this section **Shall** apply to every [USB Type-C 2.3] DRP.

8.3.3.20.1 DFP to UFP Data Role Swap State Diagram

Figure 8-177: "DFP to UFP Data Role Swap State Diagram" shows the additional state diagram required to perform a Data Role Swap from DFP to UFP operation and the changes that **Shall** be followed for error and Hard Reset handling.

Figure 8-177: "DFP to UFP Data Role Swap State Diagram"



8.3.3.20.1.1 PE_SRC_Ready or PE_SNK_Ready State

✖ The Data Role Swap process **Shall** start only from either the **PE_SRC_Ready** or **PE_SNK_Ready** state where power is stable.

The Policy Engine **Shall** transition to the **PE_DRS_DFP_UFP_Evaluate_Swap** state when:

- A **DR_Swap** Message is received and
- There are no *Active Modes* (not in Modal Operation).

The Policy Engine **Shall** transition to either the **PE_SRC_Hard_Reset** or **PE_SNK_Hard_Reset** states when:

- A **DR_Swap** Message is received and
- There are one or more *Active Modes* (Modal Operation).

The Policy Engine **Shall** transition to the **PE_DRS_DFP_UFP_Send_Swap** state when:

- The Device Policy Manager indicates that a Data Role Swap is required.

8.3.3.20.1.2 PE_DRS_DFP_UFP_Evaluate_Swap State

On entry to the **PE_DRS_DFP_UFP_Evaluate_Swap** state the Policy Engine **Shall** ask the Device Policy Manager whether a Data Role Swap can be made.

The Policy Engine **Shall** transition to the **PE_DRS_DFP_UFP_Accept_Swap** state when:

- The Device Policy Manager indicates that a Data Role Swap is ok.

The Policy Engine **Shall** transition to the **PE_DRS_DFP_UFP_Reject_Swap** state when:

- The Device Policy Manager indicates that a Data Role Swap is not ok.
- Or further evaluation of the Data Role Swap request is needed.

8.3.3.20.1.3 PE_DRS_DFP_UFP_Accept_Swap State

On entry to the **PE_DRS_DFP_UFP_Accept_Swap** state the Policy Engine **Shall** request the Protocol Layer to send an **Accept** Message.

The Policy Engine **Shall** transition to the **PE_DRS_DFP_UFP_Change_to_UFP** state when:

- The **Accept** Message has been sent.

8.3.3.20.1.4 PE_DRS_DFP_UFP_Change_to_UFP State

On entry to the **PE_DRS_DFP_UFP_Change_to_UFP** state the Policy Engine **Shall** request the Device Policy Manager to change the Port from a DFP to a UFP.

The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state when:

- The Device Policy Manager indicates that the Port has been changed to a UFP.

8.3.3.20.1.5 PE_DRS_DFP_UFP_Send_Swap State

On entry to the **PE_DRS_DFP_UFP_Send_Swap** state the Policy Engine **Shall** request the Protocol Layer to send a **DR_Swap** Message and **Shall** start the **SenderResponseTimer**.

On exit from the **PE_DRS_DFP_UFP_Send_Swap** state the Policy Engine **Shall** stop the **SenderResponseTimer**.

The Policy Engine **Shall** continue as a DFP and **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state when:

- A **Reject** Message is received.
- Or a **Wait** Message is received.
- Or the **SenderResponseTimer** times out.

The Policy Engine **Shall** transition to the **PE_DRS_DFP_UFP_Change_to_UFP** state when:

- An **Accept** Message is received.

8.3.3.20.1.6 PE_DRS_DFP_UFP_Reject_Swap State

On entry to the **PE_DRS_DFP_UFP_Reject_Swap** state the Policy Engine **Shall** request the Protocol Layer to send:

- A **Reject** Message if the device is unable to perform a Data Role Swap at this time.
- A **Wait** Message if further evaluation of the Data Role Swap request is required. Note in this case it is expected that one of the Port Partners will send a **DR_Swap** Message at a later time (see [Section 6.3.12.3 "Wait in response to a DR_Swap Message"](#)).

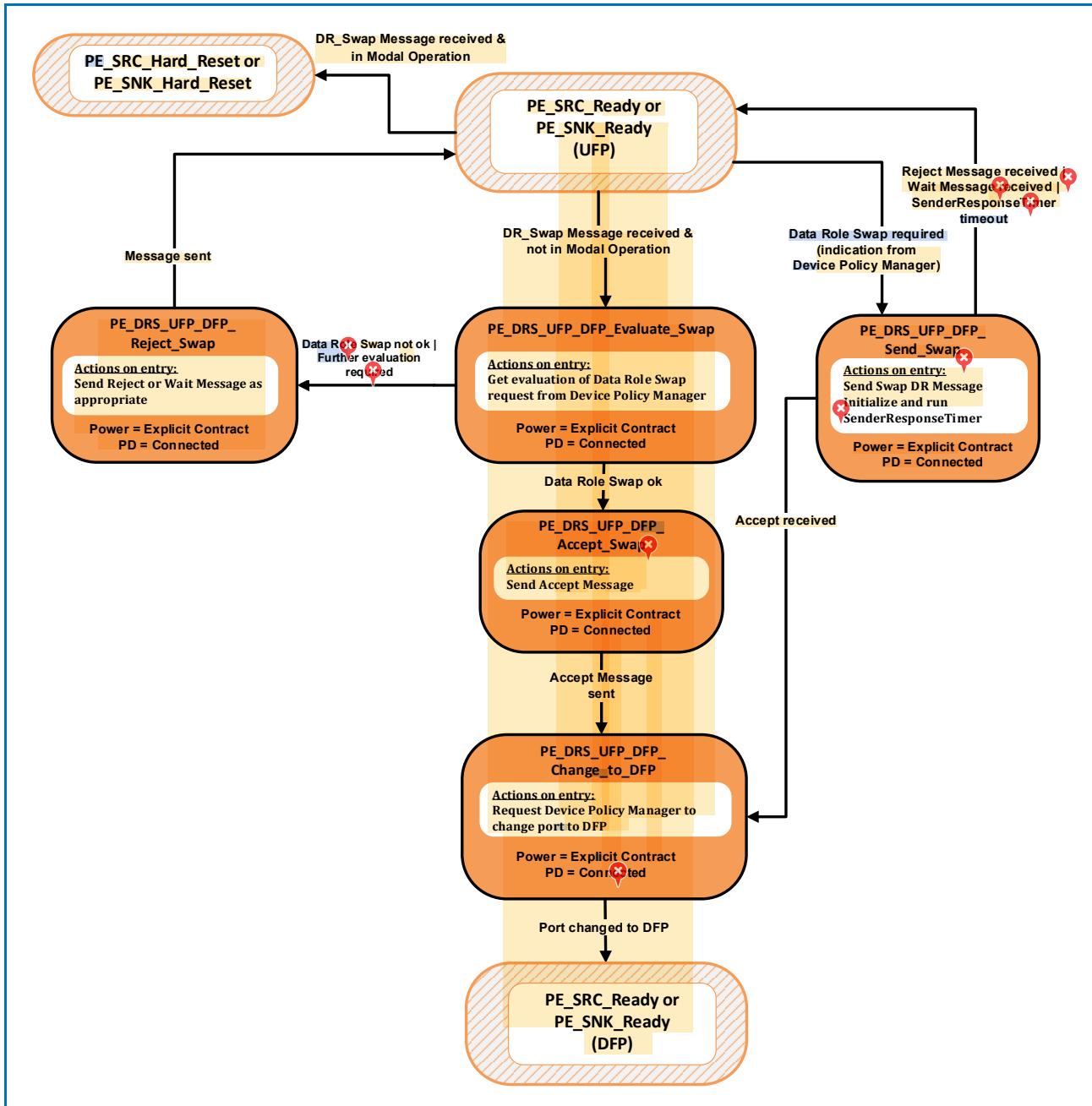
The Policy Engine **Shall** continue as a DFP and **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state when:

- The **Reject** or **Wait** Message has been sent.

8.3.3.20.2 UFP to DFP Data Role Swap State Diagram

Figure 8-178: “UFP to DFP Data Role Swap State Diagram” shows the additional state diagram required to perform a Data Role Swap from DRP UFP to DFP operation and the changes that *Shall* be followed for error and Hard Reset handling.

Figure 8-178: “UFP to DFP Data Role Swap State Diagram”



8.3.3.20.2.1 PE_SRC_Ready or PE_SNK_Ready State

The Data Role Swap process *Shall* start only from the either the **PE_SRC_Ready** or **PE_SNK_Ready** state where power is stable.

The Policy Engine **Shall** transition to the **PE_DRS_UFP_DFP_Evaluate_Swap** state when:

- A **DR_Swap** Message is received and
- There are no *Active Modes* (not in Modal Operation).

The Policy Engine **Shall** transition to either the **PE_SRC_Hard_Reset** or **PE_SNK_Hard_Reset** states when:

- A **DR_Swap** Message is received and
- There are one or more *Active Modes* (Modal Operation).

The Policy Engine **Shall** transition to the **PE_DRS_UFP_DFP_Send_Swap** state when:

- The Device Policy Manager indicates that a Data Role Swap is required.

8.3.3.20.2.2 PE_DRS_UFP_DFP_Evaluate_Swap State

On entry to the **PE_DRS_UFP_DFP_Evaluate_Swap** state the Policy Engine **Shall** ask the Device Policy Manager whether a Data Role Swap can be made.

The Policy Engine **Shall** transition to the **PE_DRS_UFP_DFP_Accept_Swap** state when:

- The Device Policy Manager indicates that a Data Role Swap is ok.

The Policy Engine **Shall** transition to the **PE_DRS_UFP_DFP_Reject_Swap** state when:

- The Device Policy Manager indicates that a Data Role Swap is not ok.
- Or further evaluation of the Data Role Swap request is needed.

8.3.3.20.2.3 PE_DRS_UFP_DFP_Accept_Swap State

On entry to the **PE_DRS_UFP_DFP_Accept_Swap** state the Policy Engine **Shall** request the Protocol Layer to send an **Accept** Message.

The Policy Engine **Shall** transition to the **PE_DRS_UFP_DFP_Change_to_DFP** state when:

- The **Accept** Message has been sent.

8.3.3.20.2.4 PE_DRS_UFP_DFP_Change_to_DFP State

On entry to the **PE_DRS_UFP_DFP_Change_to_DFP** state the Policy Engine **Shall** request the Device Policy Manager to change the Port from a UFP to a DFP.

The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state when:

- The Device Policy Manager indicates that the Port has been changed to a DFP.

8.3.3.20.2.5 PE_DRS_UFP_DFP_Send_Swap State

On entry to the **PE_DRS_UFP_DFP_Send_Swap** state the Policy Engine **Shall** request the Protocol Layer to send a **DR_Swap** Message and **Shall** start the **SenderResponseTimer**.

On exit from the **PE_DRS_UFP_DFP_Send_Swap** state the Policy Engine **Shall** stop the **SenderResponseTimer**.

The Policy Engine **Shall** continue as a UFP and **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state when:

- A **Reject** Message is received.
- Or a **Wait** Message is received.

- Or the *SenderResponseTimer* times out.

The Policy Engine ***Shall*** transition to the ***PE_DRS_UFP_DFP_Change_to_DFP*** state when:

- An *Accept* Message is received.

8.3.3.20.2.6 PE_DRS_UFP_DFP_Reject_Swap State

On entry to the ***PE_DRS_UFP_DFP_Reject_Swap*** state the Policy Engine ***Shall*** request the Protocol Layer to send:

- A *Reject* Message if the device is unable to perform a Data Role Swap at this time.
- A *Wait* Message if further evaluation of the Data Role Swap request is required. Note in this case it is expected that one of the Port Partners will send a *DR_Swap* Message at a later time (see [Section 6.3.12.3 "Wait in response to a DR_Swap Message"](#)).

The Policy Engine ***Shall*** continue as a UFP and ***Shall*** transition to the either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when:

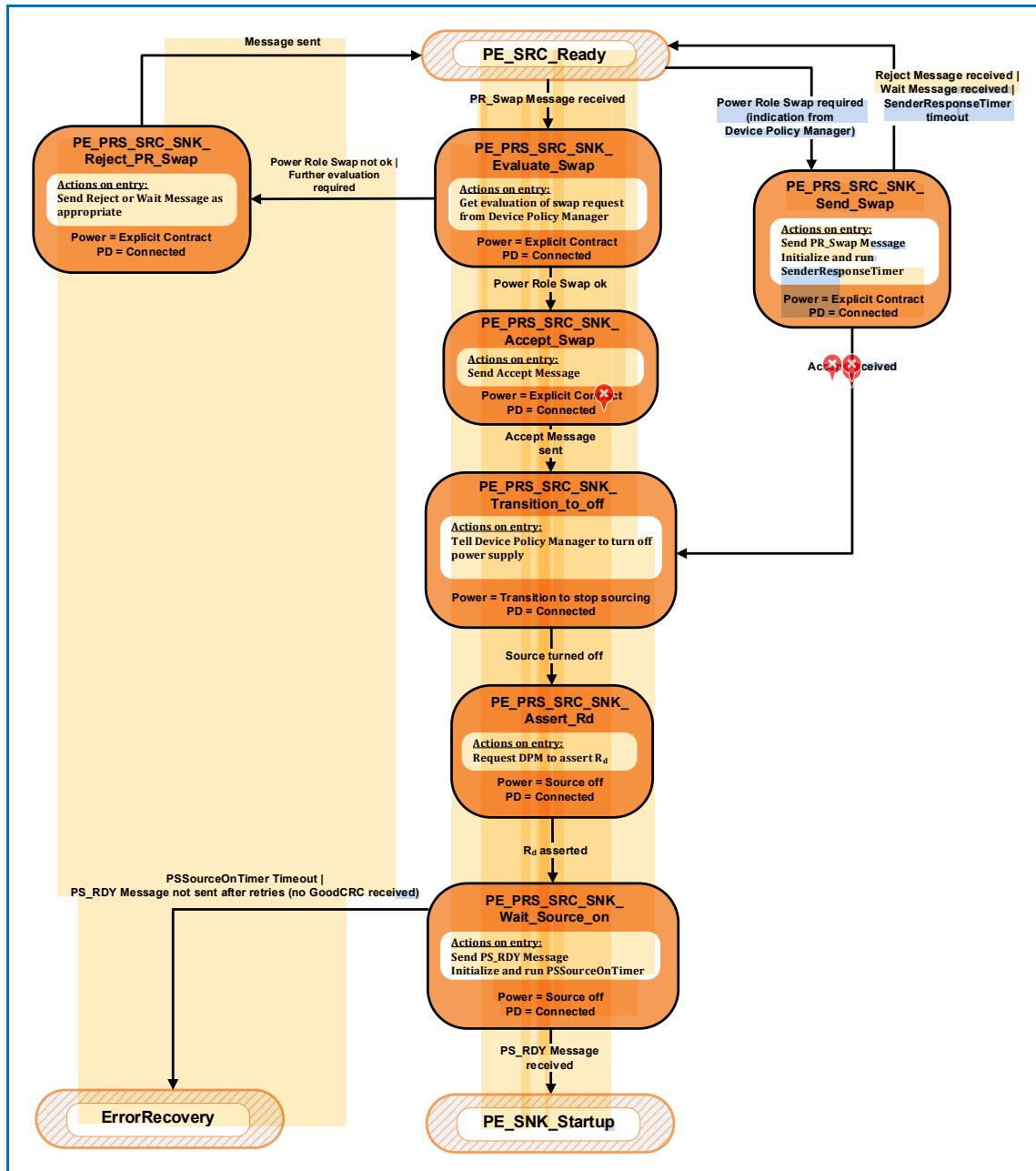
- The *Reject* or *Wait* Message has been sent.

8.3.3.20.3 Policy Engine in Source to Sink Power Role Swap State Diagram

Dual-Role Ports that combine Source and Sink capabilities **Shall** comprise Source and Sink Policy Engine state machines. In addition, they **Shall** have the capability to do a Power Role Swap from the **PE_SRC_Ready** state and **Shall** return to USB Default Operation on a Hard Reset.

Figure 8-179: "Dual-Role Port in Source to Sink Power Role Swap State Diagram" shows the additional state diagram required to perform a Power Role Swap from Source to Sink roles and the changes that **Shall** be followed for error handling.

Figure 8-179: "Dual-Role Port in Source to Sink Power Role Swap State Diagram"



8.3.3.20.3.1 PE_SRC_Ready State

The Power Role Swap process **Shall** start only from the **PE_SRC_Ready** state where power is stable.

The Policy Engine **Shall** transition to the **PE_PRS_SRC_SNK_Evaluate_Swap** state when:

- A **PR_Swap** Message is received.

The Policy Engine **Shall** transition to the **PE_PRS_SRC_SNK_Send_Swap** state when:

- The Device Policy Manager indicates that a Power Role Swap is required.

8.3.3.20.3.2 PE_PRS_SRC_SNK_Evaluate_Swap State

On entry to the **PE_PRS_SRC_SNK_Evaluate_Swap** state the Policy Engine **Shall** ask the Device Policy Manager whether a Power Role Swap can be made.

The Policy Engine **Shall** transition to the **PE_PRS_SRC_SNK_Accept_Swap** state when:

- The Device Policy Manager indicates that a Power Role Swap is ok.

The Policy Engine **Shall** transition to the **PE_PRS_SRC_SNK_Reject_Swap** state when:

- The Device Policy Manager indicates that a Power Role Swap is not ok.
- Or further evaluation of the Power Role Swap request is needed.

8.3.3.20.3.3 PE_PRS_SRC_SNK_Accept_Swap State

On entry to the **PE_PRS_SRC_SNK_Accept_Swap** state the Policy Engine **Shall** request the Protocol Layer to send an **Accept** Message.

The Policy Engine **Shall** transition to the **PE_PRS_SRC_SNK_Transition_to_off** state when:

- The **Accept** Message has been sent.

8.3.3.20.3.4 PE_PRS_SRC_SNK_Transition_to_off State

On entry to the **PE_PRS_SRC_SNK_Transition_to_off** state the Policy Engine **Shall** request the Device Policy Manager to turn off the Source.

The Policy Engine **Shall** transition to the **PE_PRS_SRC_SNK Assert_Rd** state when:

- The Device Policy Manager indicates that the Source has been turned off.

8.3.3.20.3.5 PE_PRS_SRC_SNK Assert_Rd State

On entry to the **PE_PRS_SRC_SNK Assert_Rd** state the Policy Engine **Shall** request the Device Policy Manager to change the resistor asserted on the CC wire from R_p to R_d .

The Policy Engine **Shall** transition to the **PE_PRS_SRC_SNK_Wait_Source_on** state when:

- The Device Policy Manager indicates that R_d is asserted.

8.3.3.20.3.6 PE_PRS_SRC_SNK_Wait_Source_on State

On entry to the **PE_PRS_SRC_SNK_Wait_Source_on** state the Policy Engine **Shall** request the Protocol Layer to send a **PS_RDY** Message and **Shall** start the **PSSourceOnTimer**.

On exit from the Source off state the Policy Engine **Shall** stop the **PSSourceOnTimer**.

The Policy Engine **Shall** transition to the **PE_SNK_Startup** when:

- A **PS_RDY** Message is received indicating that the remote Source is now supplying power.

The Policy Engine **Shall** transition to the **ErrorRecovery** state when:

- The **PSSourceOnTimer** times out or
- The **PS_RDY** Message is not sent after retries (a **GoodCRC** Message has not been received). **Note** a soft reset **Shall Not** be initiated in this case.

8.3.3.20.3.7 PE_PRS_SRC_SNK_Send_Swap State

On entry to the **PE_PRS_SRC_SNK_Send_Swap** state the Policy Engine **Shall** request the Protocol Layer to send a **PR_Swap** Message and **Shall** start the **SenderResponseTimer**.

On exit from the **PE_PRS_SRC_SNK_Send_Swap** state the Policy Engine **Shall** stop the **SenderResponseTimer**.

The Policy Engine **Shall** transition to the **PE_SRC_Ready** state when:

- A **Reject** Message is received.
- Or a **Wait** Message is received.
- Or the **SenderResponseTimer** times out.

The Policy Engine **Shall** transition to the **PE_PRS_SRC_SNK_Transition_to_off** state when:

- An **Accept** Message is received.

8.3.3.20.3.8 PE_PRS_SRC_SNK_Reject_Swap State

On entry to the **PE_PRS_SRC_SNK_Reject_Swap** state the Policy Engine **Shall** request the Protocol Layer to send:

- A **Reject** Message if the device is unable to perform a Power Role Swap at this time.
- A **Wait** Message if further evaluation of the Power Role Swap request is required. **Note** in this case it is expected that one of the Port Partners will send a **PR_Swap** Message at a later time (see [Section 6.3.12.3 "Wait in response to a DR_Swap Message"](#)).

The Policy Engine **Shall** transition to the **PE_SRC_Ready** when:

- The **Reject** or **Wait** Message has been sent.

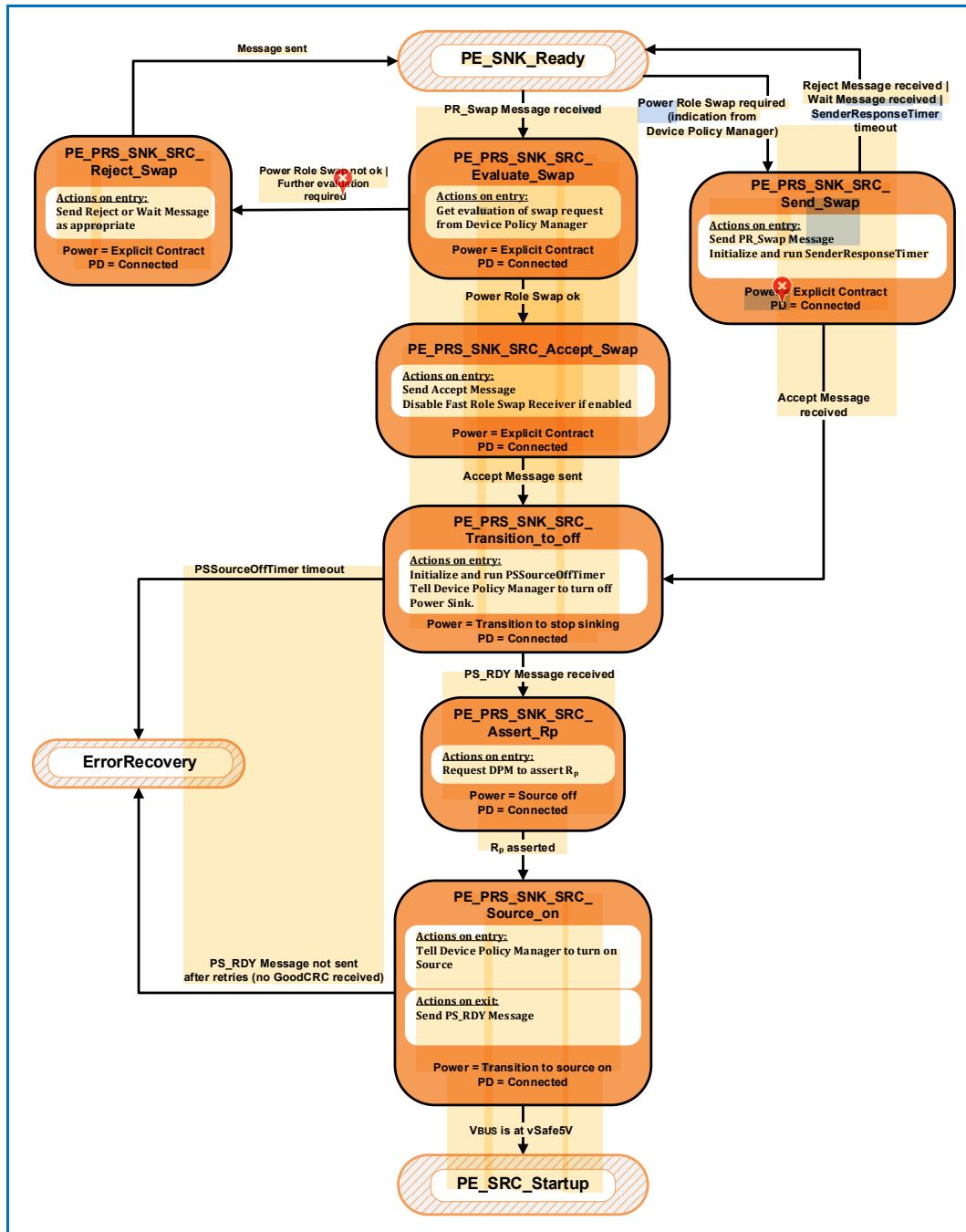
8.3.3.20.4

Policy Engine in Sink to Source Power Role Swap State Diagram

Dual-Role Ports that combine Sink and Source capabilities **Shall** comprise Sink and Source Policy Engine state machines. In addition, they **Shall** have the capability to do a Power Role Swap from the **PE_SNK_Ready** state and **Shall** return to USB Default Operation on a Hard Reset.

Figure 8-180: "Dual-role Port in Sink to Source Power Role Swap State Diagram" shows the additional state diagram required to perform a Power Role Swap from Sink to Source roles and the changes that **Shall** be followed for error handling.

Figure 8-180: "Dual-role Port in Sink to Source Power Role Swap State Diagram"



8.3.3.20.4.1 PE_SNK_Ready State

The Power Role Swap process **Shall** start only from the **PE_SNK_Ready** state where power is stable.

The Policy Engine **Shall** transition to the **PE_PRS_SNK_SRC_Evaluate_Swap** state when:

- A **PR_Swap** Message is received.

The Policy Engine **Shall** transition to the **PE_PRS_SNK_SRC_Send_Swap** state when:

- The Device Policy Manager indicates that a Power Role Swap is required.

8.3.3.20.4.2 PE_PRS_SNK_SRC_Evaluate_Swap State

On entry to the **PE_PRS_SNK_SRC_Send_Swap** state the Policy Engine **Shall** ask the Device Policy Manager whether a Power Role Swap can be made.

The Policy Engine **Shall** transition to the **PE_PRS_SNK_SRC_Accept_Swap** state when:

- The Device Policy Manager indicates that a Power Role Swap is ok.

The Policy Engine **Shall** transition to the **PE_PRS_SNK_SRC_Reject_Swap** state when:

- The Device Policy Manager indicates that a Power Role Swap is not ok.

8.3.3.20.4.3 PE_PRS_SNK_SRC_Accept_Swap State

On entry to the **PE_PRS_SNK_SRC_Accept_Swap** state the Policy Engine **Shall** request the Protocol Layer to send an **Accept** Message and **Shall** disable the Fast Role Swap receiver if this is enabled.

The Policy Engine **Shall** transition to the **PE_PRS_SNK_SRC_Transition_to_off** state when:

- The **Accept** Message has been sent.

8.3.3.20.4.4 PE_PRS_SNK_SRC_Transition_to_off State

On entry to the **PE_PRS_SNK_SRC_Transition_to_off** state the Policy Engine **Shall** initialize and run the **PSSourceOffTimer** and then request the Device Policy Manager to turn off the Sink.

The Policy Engine **Shall** transition to the **ErrorRecovery** state when:

- The **PSSourceOffTimer** times out.

The Policy Engine **Shall** transition to the **PE_PRS_SNK_SRC_Assert_Rp** state when:

- A **PS_RDY** Message is received.

8.3.3.20.4.5 PE_PRS_SNK_SRC_Assert_Rp State

On entry to the **PE_PRS_SNK_SRC_Assert_Rp** state the Policy Engine **Shall** request the Device Policy Manager to change the resistor asserted on the CC wire from R_d to R_p .

The Policy Engine **Shall** transition to the **PE_PRS_SNK_SRC_Source_on** state when:

- The Device Policy Manager indicates that R_d is asserted.

8.3.3.20.4.6 PE_PRS_SNK_SRC_Source_on State

On entry to the **PE_PRS_SNK_SRC_Source_on** state the Policy Engine **Shall** request the Device Policy Manager to turn on the Source.

On exit from the ***PE_PRS_SNK_SRC_Source_on*** state the Policy Engine ***Shall*** send a ***PS_RDY*** Message.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Startup*** state when:

- The Source Port V_{BUS} is at ***vSafe5V***.

The Policy Engine ***Shall*** transition to the ***ErrorRecovery*** state when:

- The ***PS_RDY*** Message is not sent after retries (a ***GoodCRC*** Message has not been received). A soft reset ***Shall Not*** be initiated in this case.

8.3.3.20.4.7 PE_PRS_SNK_SRC_Send_Swap State

On entry to the ***PE_PRS_SNK_SRC_Send_Swap*** state the Policy Engine ***Shall*** request the Protocol Layer to send a ***PR_Swap*** Message and ***Shall*** initialize and run the ***SenderResponseTimer***.

The Policy Engine ***Shall*** transition to the ***PE_SNK_Ready*** state when:

- A ***Reject*** Message is received.
- Or a ***Wait*** Message is received.
- Or the ***SenderResponseTimer*** times out.

The Policy Engine ***Shall*** transition to the ***PE_PRS_SNK_SRC_Transition_to_off*** state when:

- An ***Accept*** Message is received.

8.3.3.20.4.8 PE_PRS_SNK_SRC_Reject_Swap State

On entry to the ***PE_PRS_SNK_SRC_Reject_Swap*** state the Policy Engine ***Shall*** request the Protocol Layer to send:

- A ***Reject*** Message if the device is unable to perform a Power Role Swap at this time.
- A ***Wait*** Message if further evaluation of the Power Role Swap request is required. Note in this case it is expected that one of the Port Partners will send a ***PR_Swap*** Message at a later time (see ***Section 6.3.12.3 "Wait in response to a DR_Swap Message"***).

The Policy Engine ***Shall*** transition to the ***PE_SNK_Ready*** state when:

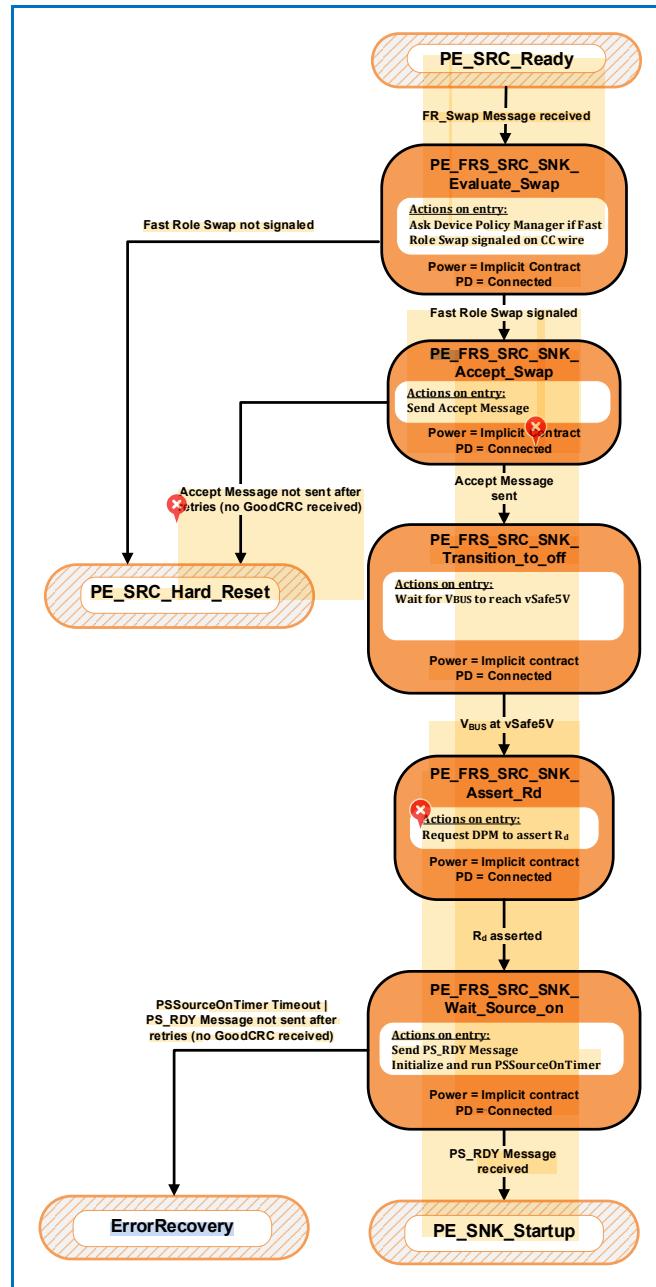
- The ***Reject*** or ***Wait*** Message has been sent.

8.3.3.20.5 Policy Engine in Source to Sink Fast Role Swap State Diagram

Dual-Role Ports that combine Source and Sink capabilities **Shall** comprise Source and Sink Policy Engine state machines. In addition, they **Should** have the capability to do a Fast Role Swap from the **PE_SRC_Ready** state and **Shall** return to USB Default Operation on a Hard Reset.

Figure 8-181: “Dual-Role Port in Source to Sink Fast Role Swap State Diagram” shows the additional state diagram required to perform a Fast Role Swap from Source to Sink roles and the changes that **Shall** be followed for error handling.

Figure 8-181: “Dual-Role Port in Source to Sink Fast Role Swap State Diagram”



8.3.3.20.5.1 PE_SRC_Ready State

The Fast Role Swap process **Shall** start only from the **PE_SRC_Ready** state where power is stable.

The Policy Engine **Shall** transition to the **PE_FRS_SRC_SNK_Evaluate_Swap** state when:

- An **FR_Swap** Message is received.

8.3.3.20.5.2 PE_FRS_SRC_SNK_Evaluate_Swap State

On entry to the **PE_FRS_SRC_SNK_Evaluate_Swap** state the Policy Engine **Shall** ask the Device Policy Manager whether Fast Role Swap has been signaled on the CC wire.

The Policy Engine **Shall** transition to the **PE_FRS_SRC_SNK_Accept_Swap** state when:

- The Device Policy Manager indicates that a Fast Role Swap has been signaled.

The Policy Engine **Shall** transition to the **PE_SRC_Hard_Reset** state when:

- The Device Policy Manager indicates that a Fast Role Swap is not being signaled.

8.3.3.20.5.3 PE_FRS_SRC_SNK_Accept_Swap State

On entry to the **PE_FRS_SRC_SNK_Accept_Swap** state the Policy Engine **Shall** request the Protocol Layer to send an **Accept** Message.

The Policy Engine **Shall** transition to the **PE_FRS_SNK_SRC_Transition_to_off** state when:

- The **Accept** Message has been sent.

The Policy Engine **Shall** transition to the **PE_SRC_Hard_Reset** state when:

- The **Accept** Message is not sent after retries (a **GoodCRC** Message has not been received). Note a soft reset **Shall Not** be initiated in this case.

8.3.3.20.5.4 PE_FRS_SRC_SNK_Transition_to_off State

On entry to the **PE_FRS_SNK_SRC_Transition_to_off** state the Policy Engine **Shall** wait until V_{BUS} has discharged to **vSafe5V**.

The Policy Engine **Shall** transition to the **PE_PRS_SRC_SNK Assert_Rd** state when:

- The Device Policy Manager indicates that V_{BUS} has discharged to **vSafe5V**.

8.3.3.20.5.5 PE_FRS_SRC_SNK Assert_Rd State

On entry to the **PE_PRS_SRC_SNK Assert_Rd** state the Policy Engine **Shall** request the Device Policy Manager to change the resistor asserted on the CC wire from R_p to R_d.

The Policy Engine **Shall** transition to the **PE_PRS_SRC_SNK_Wait_Source_on** state when:

- The Device Policy Manager indicates that R_d is asserted.

8.3.3.20.5.6 PE_FRS_SRC_SNK_Wait_Source_on State

On entry to the **PE_PRS_SRC_SNK_Wait_Source_on** state the Policy Engine **Shall** request the Protocol Layer to send a **PS_RDY** Message and **Shall** start the **PSSourceOnTimer**.

On exit from the Source off state the Policy Engine **Shall** stop the **PSSourceOnTimer**.

The Policy Engine **Shall** transition to the **PE_SNK_Startup** when:

- A ***PS_RDY*** Message is received indicating that the new Source is now applying R_p.
- ✖ The Policy Engine ***Shall*** transition to the ***ErrorRecovery*** state when:
- The ***PSSourceOnTimer*** times out or
 - The ***PS_RDY*** Message is not sent after retries (a ***GoodCRC*** Message has not been received). **Note** a soft reset ***Shall Not*** be initiated in this case.

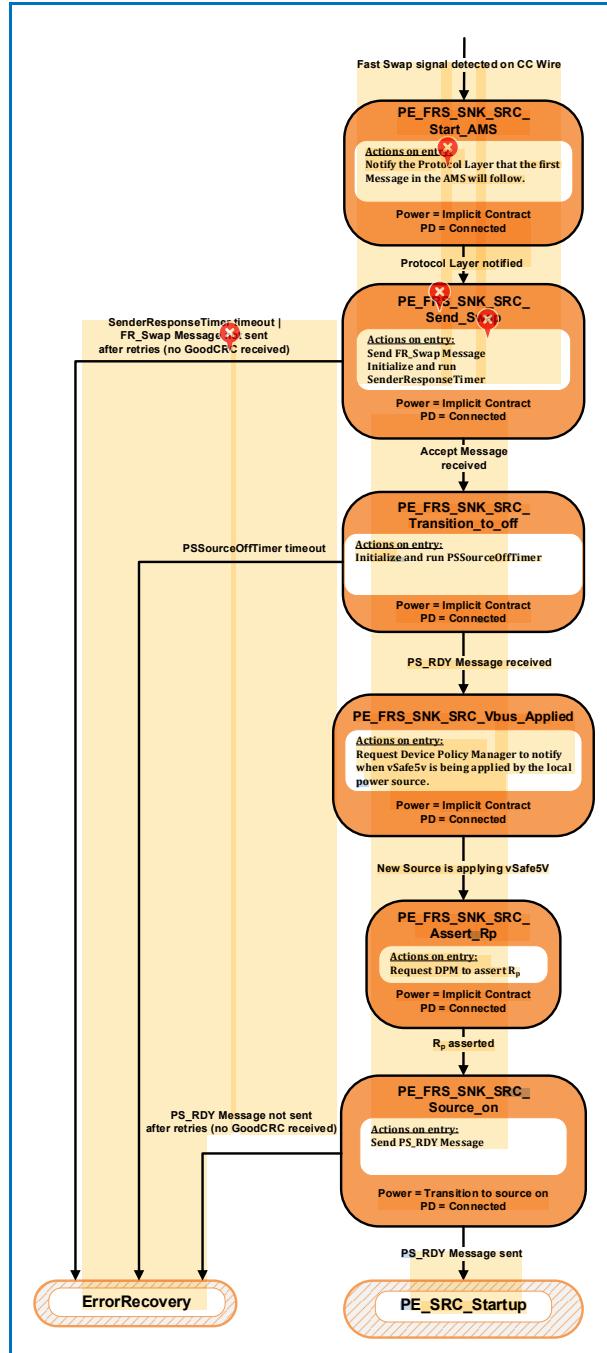
8.3.3.20.6

Policy Engine in Sink to Source Fast Role Swap State Diagram

Dual-Role Ports that combine Sink and Source capabilities **Shall** comprise Sink and Source Policy Engine state machines. In addition, they **Should** have the capability to do a Fast Role Swap from the **PE_SNK_Ready** state and **Shall** return to USB Default Operation on a Hard Reset.

Figure 8-182: “Dual-role Port in Sink to Source Fast Role Swap State Diagram” shows the additional state diagram required to perform a Fast Role Swap from Sink to Source roles and the changes that **Shall** be followed for error handling.

Figure 8-182: “Dual-role Port in Sink to Source Fast Role Swap State Diagram”



8.3.3.20.6.1 PE_FRS_SNK_SRC_Start_AMS State

The Policy Engine **Shall** transition to the **PE_FRS_SNK_SRC_Send_Swap** state from any other state provided there is an Explicit Contract in place when:

- The Sink Capabilities received from the initial Source by the Policy Engine has at least one of the Fast Role Swap bits set.
- The system has sufficient reserve power to provide the requested current to the initial Source, as requested in the Fast Role Swap bits in the Sink Capabilities, and is willing to dedicate it to the Port
- The Device Policy Manager indicates that a Fast Role Swap signal has been detected on the CC Wire.

On entry to the **PE_FRS_SNK_SRC_Start_AMS** state the Policy Engine **Shall** notify the Protocol Layer that the first Message in an AMS will follow.

The Policy Engine **Shall** transition to the **PE_FRS_SNK_SRC_Send_Swap** state when:

- The Protocol Layer has been notified.

8.3.3.20.6.2 PE_FRS_SNK_SRC_Send_Swap State

On entry to the **PE_FRS_SNK_SRC_Send_Swap** state the Policy Engine **Shall** request the Protocol Layer to send an **FR_Swap** Message and **Shall** initialize and run the **SenderResponseTimer**.

The Policy Engine **Shall** transition to the **PE_FRS_SNK_SRC_Transition_to_off** state when:

- An **Accept** Message is received.

The Policy Engine **Shall** transition to the **ErrorRecovery** state when:

- The **SenderResponseTimer** times out or
- The **FR_Swap** Message is not sent after retries (a **GoodCRC** Message has not been received). A soft reset **Shall Not** be initiated in this case.

8.3.3.20.6.3 PE_FRS_SNK_SRC_Transition_to_off State

On entry to the **PE_FRS_SNK_SRC_Transition_to_off** state the Policy Engine **Shall** initialize and run the **PSSourceOffTimer** and then request the Device Policy Manager to turn off the Sink.

The Policy Engine **Shall** transition to the **ErrorRecovery** state when:

- The **PSSourceOffTimer** times out.

The Policy Engine **Shall** transition to the **PE_FRS_SNK_SRC_Vbus_Applied** state when:

- A **PS_RDY** Message is received.

8.3.3.20.6.4 PE_FRS_SNK_SRC_Vbus_Applied State

On entry to the **PE_FRS_SNK_SRC_Vbus_Applied** state the Policy Engine waits for a notification from the Device Policy Manager that the local power source has applied **vSafe5V** to VBUS (see [Section 5.8.6.3 "Fast Role Swap Detection"](#)). Note this could have already been applied prior to entering this state or could be applied while waiting in this state.

The Policy Engine **Shall** transition to the **PE_FRS_SNK_SRC Assert_Rp** state when:

- The Device Policy Manager indicates that **vSafe5V** is being applied.

8.3.3.20.6.5 PE_FRS_SNK_SRC Assert Rp State

On entry to the **PE_FRS_SNK_SRC Assert Rp** state the Policy Engine **Shall** request the Device Policy Manager to change the resistor asserted on the CC wire from R_d to R_p .

The Policy Engine **Shall** transition to the **PE_FRS_SNK_SRC Source on** state when:

- The Device Policy Manager indicates that R_p is asserted.

8.3.3.20.6.6 PE_FRS_SNK_SRC Source on State

On entry to the **PE_FRS_SNK_SRC Source on** state the Policy Engine **Shall** request the Device Policy Manager to turn on the Source.

On exit from the **PE_FRS_SNK_SRC Source on** state (except if the exit is to send a **Ping** Message) the Policy Engine **Shall** send a **PS_RDY** Message.

The Policy Engine **Shall** transition to the **PE_SRC_Startup** state when:

- The **PS_RDY** Message has been sent.

The Policy Engine **Shall** transition to the **ErrorRecovery** state when:

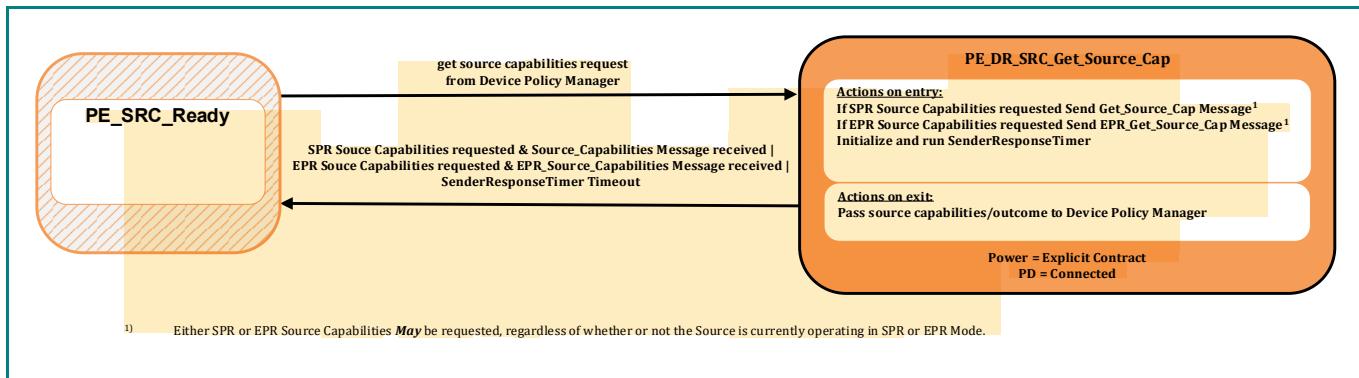
- The **PS_RDY** Message is not sent after retries (a **GoodCRC** Message has not been received). A soft reset **Shall Not** be initiated in this case.

8.3.3.20.7

Dual-Role (Source Port) Get Source Capabilities State Diagram

Figure 8-183 “Dual-Role (Source) Get Source Capabilities diagram” shows the state diagram for a Dual-Role device, presently operating as a Source, on receiving a request from the Device Policy Manager to get the Port Partner’s Source capabilities. See also [Section 6.4.1.1.3 “Use by Dual-Role Power devices”](#).

Figure 8-183 “Dual-Role (Source) Get Source Capabilities diagram”



8.3.3.20.7.1

PE_DR_SRC_Get_Source_Cap State

The Policy Engine **Shall** transition to the **PE_DR_SRC_Get_Source_Cap** state, from the **PE_SRC_Ready** state, due to a request to get the remote source capabilities from the Device Policy Manager.

- On entry to the **PE_DR_SRC_Get_Source_Cap** state the Policy Engine **Shall** request the Protocol Layer to send a get Source Capabilities message in order to retrieve the Source’s capabilities. The Policy Engine **Shall** send:
 - A **Get_Source_Cap** Message when the Device Policy Manager requests SPR capabilities or
 - An **EPR_Get_Source_Cap** Message when the Device Policy Manager requests EPR Capabilities.

The Policy Engine **Shall** then start the **SenderResponseTimer**.

On exit from the **PE_DR_SRC_Get_Source_Cap** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (capabilities or response timeout).

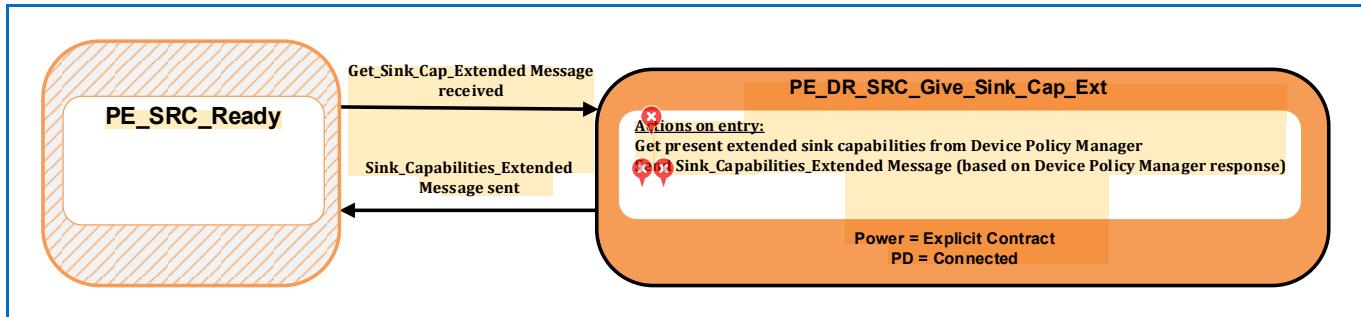
The Policy Engine **Shall** transition back to the **PE_SRC_Ready** State (see [Figure 8-134 “Source Port State Diagram”](#)) when:

- In SPR Mode and SPR Source Capabilities were requested and a **Source_Capabilities** Message is received or
- In EPR Mode and EPR Source Capabilities were requested and an **EPR_Source_Capabilities** Message is received or
- The **SenderResponseTimer** times out.

8.3.3.20.8 Dual-Role (Source Port) Give Sink Capabilities State Diagram

Figure 8-184 “Dual-Role (Source) Give Sink Capabilities diagram” shows the state diagram for a Dual-Role device, presently operating as a Source, on receiving a *Get_Sink_Cap* Message. See also *Section 6.4.1.1.3 “Use by Dual-Role Power devices”*.

Figure 8-184 “Dual-Role (Source) Give Sink Capabilities diagram”



8.3.3.20.8.1 PE_DR_SRC_Give_Sink_Cap State

The Policy Engine **Shall** transition to the **PE_DR_SRC_Give_Sink_Cap** state, from the **PE_SRC_Ready** state, when a *Get_Sink_Cap* Message or *EPR_Get_Sink_Cap* Message is received.

- On entry to the **PE_DR_SRC_Give_Sink_Cap** state the Policy Engine **Shall** request the Device Policy Manager for the current system capabilities. The Policy Engine **Shall** then request the Protocol Layer to send a *Sink_Capabilities* Message containing these capabilities. The Policy Engine **Shall** send:
 - A *Sink_Capabilities* Message when a *Get_Sink_Cap* Message is received or
 - An *EPR_Sink_Capabilities* Message when a *EPR_Get_Sink_Cap* Message is received.

The Policy Engine **Shall** transition back to the **PE_SRC_Ready** state (see *Figure 8-134 “Source Port State Diagram”*) when:

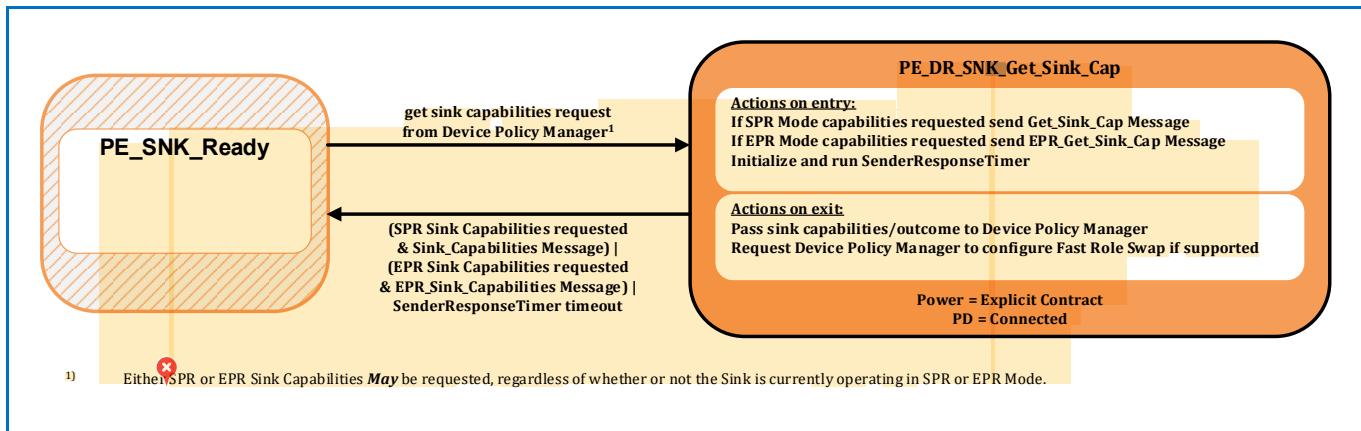
- The Sink Capabilities Message has been successfully sent.

8.3.3.20.9

Dual-Role (Sink Port) Get Sink Capabilities State Diagram

Figure 8-185 “Dual-Role (Sink) Get Sink Capabilities State Diagram” shows the state diagram for a Dual-Role device, presently operating as a Sink, on receiving a request from the Device Policy Manager to get the Port Partner’s Sink capabilities. See also [Section 6.4.1.1.3 “Use by Dual-Role Power devices”](#).

Figure 8-185 “Dual-Role (Sink) Get Sink Capabilities State Diagram”



8.3.3.20.9.1

PE_DR_SNK_Get_Sink_Cap State

The Policy Engine **Shall** transition to the **PE_DR_SNK_Get_Sink_Cap** state, from the **PE_SNK_Ready** state, due to a request to get the remote source capabilities from the Device Policy Manager.

- On entry to the **PE_DR_SNK_Get_Sink_Cap** state the Policy Engine **Shall** request the Protocol Layer to send a get Sink Capabilities message in order to retrieve the Sink’s capabilities. The Policy Engine **Shall** send:
 - A **Get_Sink_Cap** Message when the Device Policy Manager requests SPR capabilities or
 - An **EPR_Get_Sink_Cap** Message when the Device Policy Manager requests EPR Capabilities.

The Policy Engine **Shall** then start the **SenderResponseTimer**.

On exit from the **PE_SRC_Get_Sink_Cap** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (capabilities or response timeout). If Fast Role Swap is supported, request Device Policy Manager prepare or disable 5V source and configure the Fast Role Swap receiver based on the Fast Role Swap bits in the received Sink Capabilities.

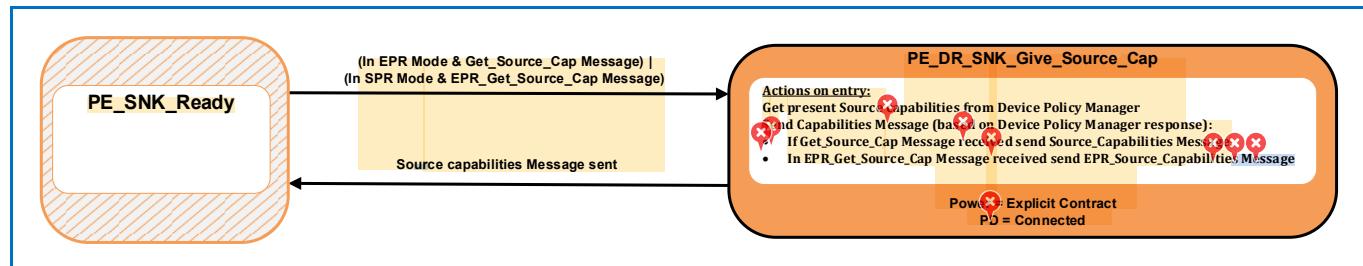
The Policy Engine **Shall** transition to the **PE_SNK_Ready** state (see [Figure 8-135 “Sink Port State Diagram”](#)) when:

- SPR Sink Capabilities were requested and a **Sink_Capabilities** Message is received or
- EPR Sink Capabilities were requested and an **EPR_Sink_Capabilities** Message is received or
- The **SenderResponseTimer** times out.

8.3.3.20.10 Dual-Role (Sink Port) Give Source Capabilities State Diagram

Figure 8-186 “Dual-Role (Sink) Give Source Capabilities State Diagram” shows the state diagram for a Dual-Role device, presently operating as a Sink, on receiving a *Get_Source_Cap* Message. See also [Section 6.4.1.1.3 “Use by Dual-Role Power devices”](#).

Figure 8-186 “Dual-Role (Sink) Give Source Capabilities State Diagram”



8.3.3.20.10.1 PE_DR_SNK_Give_Source_Cap State

The Policy Engine **Shall** transition to the **PE_DR_SNK_Give_Source_Cap** state, from the **PE_SNK_Ready** state, when a *Get_Source_Cap* Message is received.

- On entry to the **PE_DR_SNK_Give_Source_Cap** State the Policy Engine **Shall** request the Device Policy Manager for the current system capabilities. The Policy Engine **Shall** then request the Protocol Layer to send a Source Capabilities Message containing these capabilities.
- The Policy Engine **Shall** send:
 - A *Source_Capabilities* Message when a *Get_Source_Cap* Message is received or
 - An *EPR_Source_Capabilities* Message when a *EPR_Get_Source_Cap* Message is received.

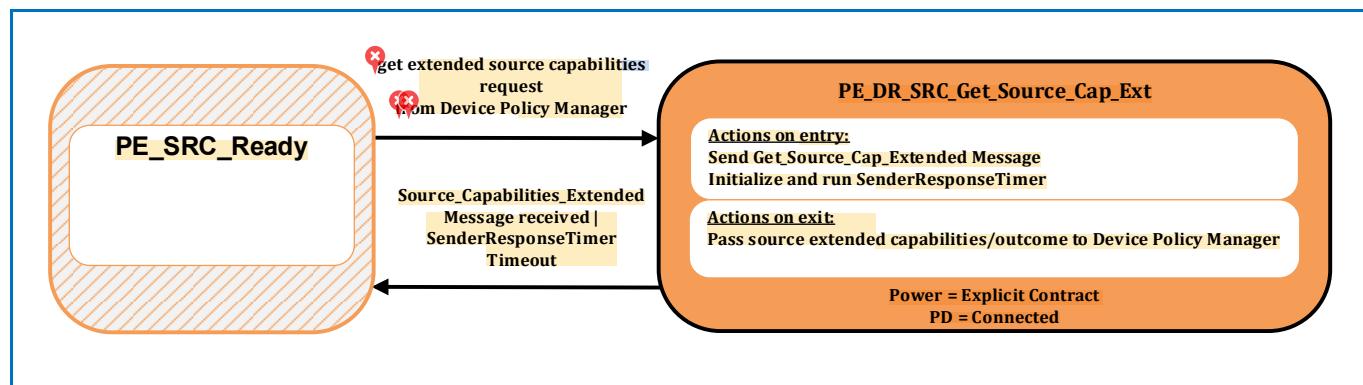
The Policy Engine **Shall** transition to the **PE_SNK_Ready** state (see [Figure 8-135 “Sink Port State Diagram”](#)) when:

- The Source Capabilities Message has been successfully sent.

8.3.3.20.11 Dual-Role (Source Port) Get Source Capabilities Extended State Diagram

Figure 8-187 “Dual-Role (Source) Get Source Capabilities Extended State Diagram” shows the state diagram for a Dual-Role device, presently operating as a Source, on receiving a request from the Device Policy Manager to get the Port Partner’s extended Source capabilities. See also [Section 6.5.1 “Source_Capabilities_Extended Message”](#).

Figure 8-187 “Dual-Role (Source) Get Source Capabilities Extended State Diagram”



8.3.3.20.11.1 PE_DR_SRC_Get_Source_Cap_Ext State

The Policy Engine **Shall** transition to the **PE_DR_SRC_Get_Source_Cap_Ext** state, from the **PE_SRC_Ready** state, due to a request to get the remote extended source capabilities from the Device Policy Manager.

On entry to the **PE_DR_SRC_Get_Source_Cap_Ext** state the Policy Engine **Shall** send a **Get_Source_Cap_Extended** Message and initialize and run the **SenderResponseTimer**.

On exit from the **PE_DR_SRC_Get_Source_Cap_Ext** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (capabilities or response timeout).

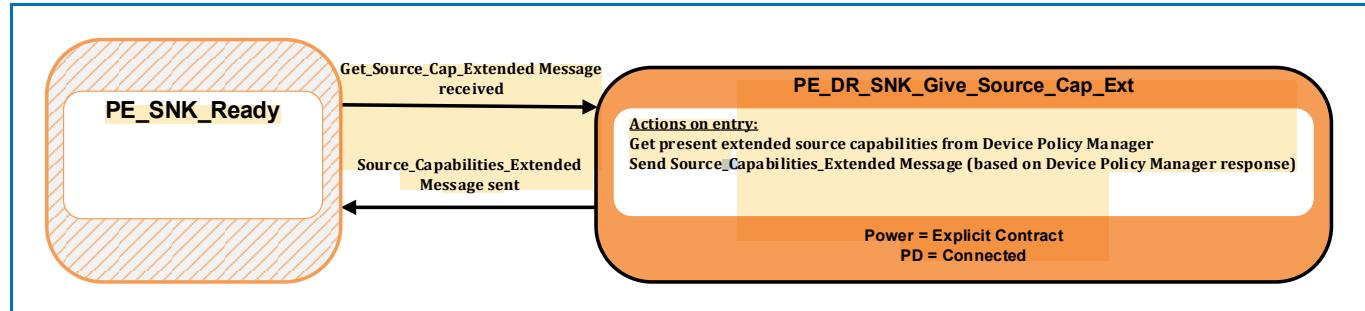
The Policy Engine **Shall** transition back to the **PE_SRC_Ready** state (see [Figure 8-134 “Source Port State Diagram”](#)) when:

- A **Source_Capabilities_Extended** Message is received
- Or **SenderResponseTimer** times out.

8.3.3.20.12 Dual-Role (Sink Port) Give Source Capabilities Extended State Diagram

Figure 8-188 “Dual-Role (Sink) Give Source Capabilities Extended diagram” shows the state diagram for a Dual-Role device, presently operating as a Sink, on receiving a *Get_Source_Cap_Extended* Message. See also [Section 6.5.1 “Source_Capabilities_Extended Message”](#).

Figure 8-188 “Dual-Role (Sink) Give Source Capabilities Extended diagram”



8.3.3.20.12.1 PE_DR_SNK_Give_Source_Cap_Ext State

The Policy Engine **Shall** transition to the **PE_DR_SNK_Give_Source_Cap_Ext** state, from the **PE_SNK_Ready** state, when a *Get_Source_Cap_Extended* Message is received.

On entry to the **PE_DR_SNK_Give_Source_Cap_Ext** state the Policy Engine **Shall** request the present extended Source capabilities from the Device Policy Manager and then send a *Source_Capabilities_Extended* Message based on these capabilities.

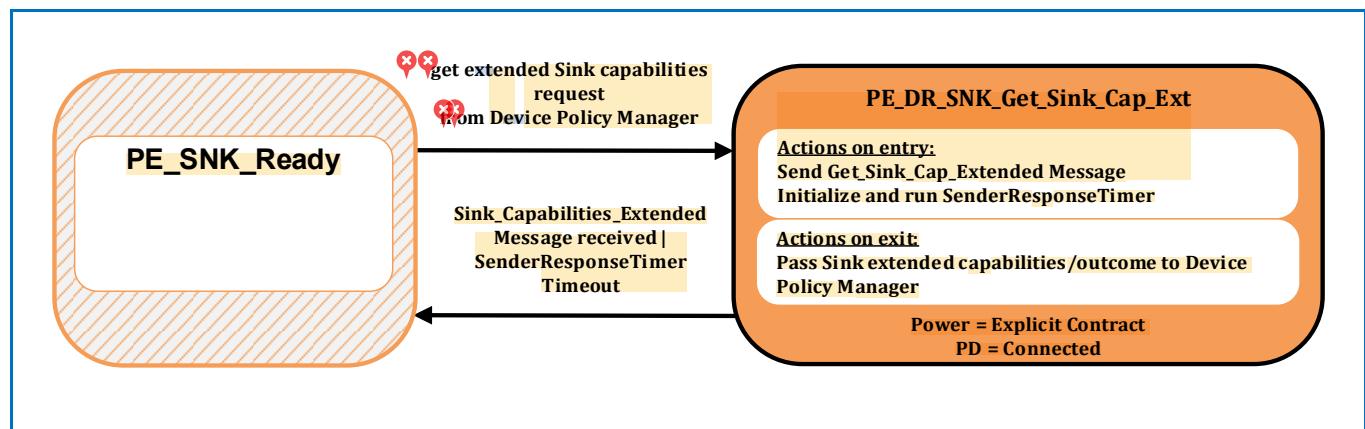
The Policy Engine **Shall** transition back to the **PE_SNK_Ready** state (see [Figure 8-135 “Sink Port State Diagram”](#)) when:

- The *Source_Capabilities_Extended* Message has been successfully sent.

8.3.3.20.13 Dual-Role (Sink Port) Get Sink Capabilities Extended State Diagram

Figure 8-189 “Dual-Role (Sink) Get Sink Capabilities Extended State Diagram” shows the state diagram for a Dual-Role device, presently operating as a Sink, on receiving a request from the Device Policy Manager to get the Port Partner’s extended Sink capabilities. See also [Section 6.5.13 “Sink_Capabilities_Extended Message”](#).

Figure 8-189 “Dual-Role (Sink) Get Sink Capabilities Extended State Diagram”



8.3.3.20.13.1 PE_DR_SNK_Get_Sink_Cap_Ext State

The Policy Engine **Shall** transition to the **PE_DR_SNK_Get_Sink_Cap_Ext** state, from the **PE_SNK_Ready** state, due to a request to get the remote extended source capabilities from the Device Policy Manager.

On entry to the **PE_DR_SNK_Get_Sink_Cap_Ext** state the Policy Engine **Shall** send a **Get_Sink_Cap_Extended** Message and initialize and run the **SenderResponseTimer**.

On exit from the **PE_DR_SNK_Get_Sink_Cap_Ext** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (capabilities or response timeout).

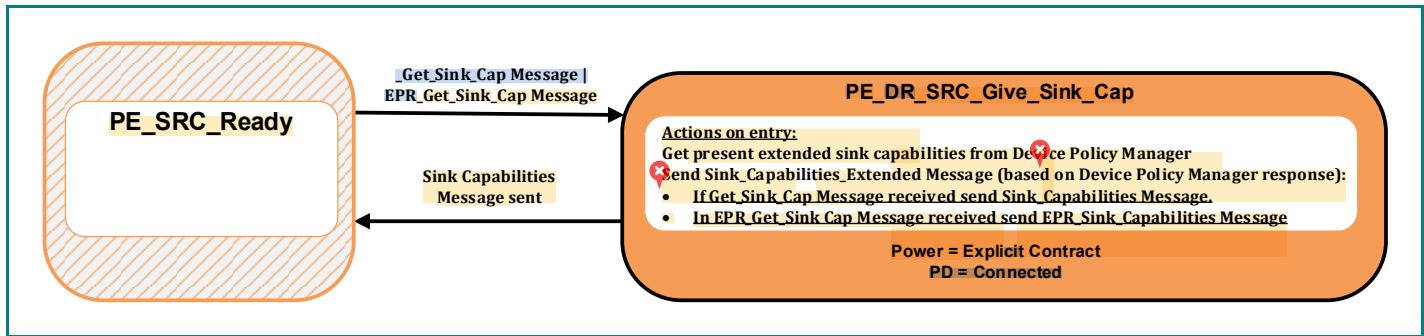
The Policy Engine **Shall** transition back to the **PE_SNK_Ready** state (see [Figure 8-135 “Sink Port State Diagram”](#)) when:

- A **Sink_Capabilities_Extended** Message is received.
- Or **SenderResponseTimer** times out.

8.3.3.20.14 Dual-Role (Source Port) Give Sink Capabilities Extended State Diagram

[Figure 8-190 “Dual-Role \(Source\) Give Sink Capabilities Extended diagram”](#) shows the state diagram for a Dual-Role device, presently operating as a Sink, on receiving a **Get_Sink_Cap_Extended** Message. See also [Section 6.5.13 “Sink_Capabilities_Extended Message”](#).

[Figure 8-190 “Dual-Role \(Source\) Give Sink Capabilities Extended diagram”](#)



8.3.3.20.14.1 PE_DR_SNK_Give_Source_Cap_Ext State

The Policy Engine **Shall** transition to the **PE_DR_SRC_Give_Sink_Cap_Ext** state, from the **PE_SRC_Ready** state, when a **Get_Sink_Cap_Extended** Message is received.

On entry to the **PE_DR_SRC_Give_Sink_Cap_Ext** state the Policy Engine **Shall** request the present extended Sink capabilities from the Device Policy Manager and then send a **Sink_Capabilities_Extended** Message based on these capabilities.

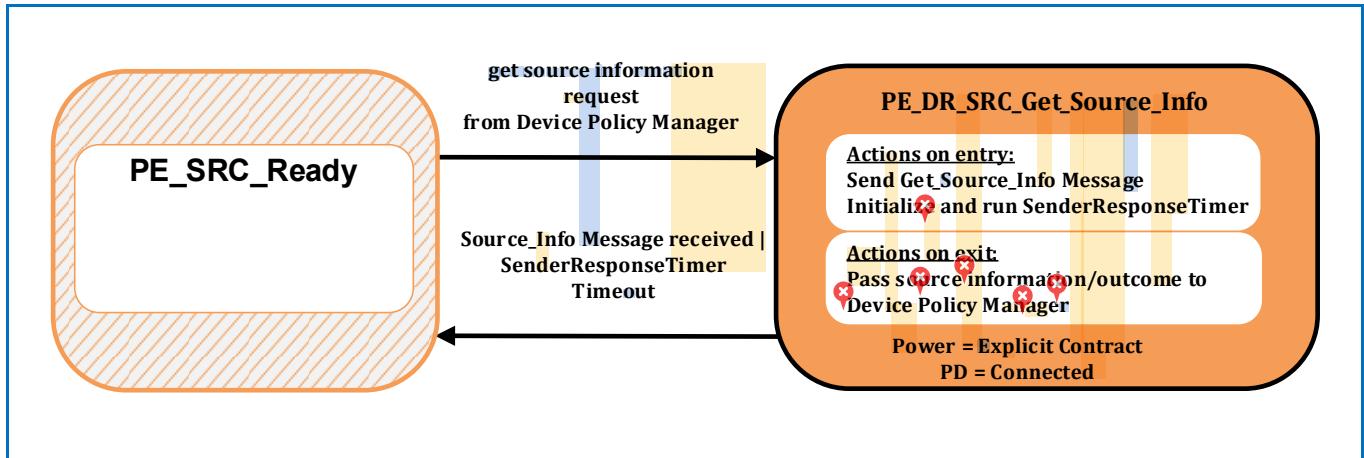
The Policy Engine **Shall** transition back to the **PE_SRC_Ready** state (see [Figure 8-134 “Source Port State Diagram”](#)) when:

- The **Sink_Capabilities_Extended** Message has been successfully sent.

8.3.3.20.15 Dual-Role (Source Port) Get Source Information State Diagram

Figure 8-191 “Dual-Role (Source) Get Source Information State Diagram” shows the state diagram for a Dual-Role device, presently operating as a Source, on receiving a request from the Device Policy Manager to get the Port Partner’s Source information. See also [Section 6.3.23 “Get_Source_Info Message”](#) and [Section 6.4.11 “Source_Info Message”](#).

Figure 8-191 “Dual-Role (Source) Get Source Information State Diagram”



8.3.3.20.15.1 PE_DR_SRC_Get_Source_Info State

The Policy Engine **Shall** transition to the **PE_DR_SRC_Get_Source_Info** state, from the **PE_SRC_Ready** state, due to a request to get the remote source information from the Device Policy Manager.

On entry to the **PE_DR_SRC_Get_Source_Info** state the Policy Engine **Shall** send a **Get_Source_Info** Message and initialize and run the **SenderResponseTimer**.

On exit from the **PE_DR_SRC_Get_Source_Info** state the Policy Engine **Shall** inform the Device Policy Manager of the outcome (information or response timeout).

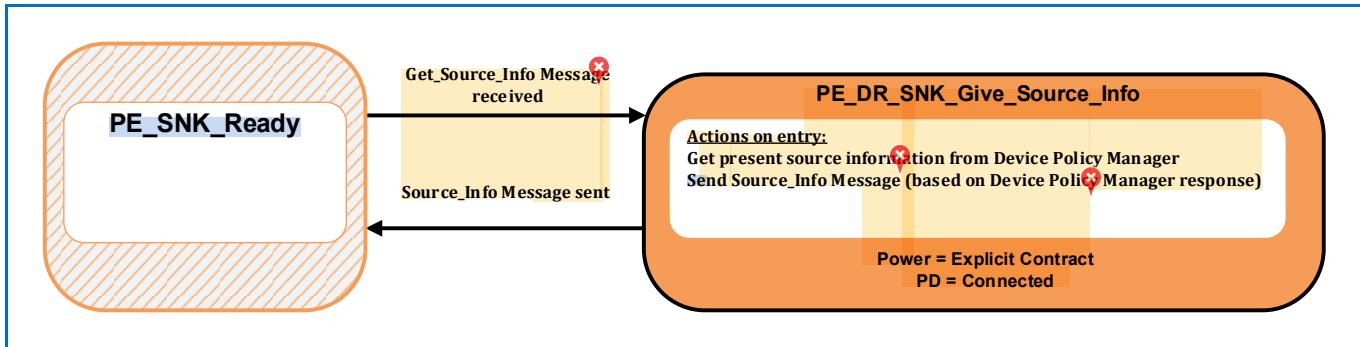
The Policy Engine **Shall** transition back to the **PE_SRC_Ready** state (see [Figure 8-134 “Source Port State Diagram”](#)) when:

- A **Source_Info** Message is received.
- Or **SenderResponseTimer** times out.

8.3.3.20.16 Dual-Role (Sink Port) Give Source Information State Diagram

Figure 8-192 “Dual-Role (Source) Give Source Information diagram” shows the state diagram for a Dual-Role device, presently operating as a Sink, on receiving a **Get_Source_Info** Message. See also [Section 6.3.23 “Get_Source_Info Message”](#) and [Section 6.4.11 “Source_Info Message”](#).

Figure 8-192 “Dual-Role (Source) Give Source Information diagram”



8.3.3.20.16.1 PE_DR_SNK_Give_Source_Info State

The Policy Engine **Shall** transition to the **PE_DR_SNK_Give_Source_Info** state, from the **PE_SNK_Ready** state, when a **Get_Source_Info** Message is received.

On entry to the **PE_DR_SNK_Give_Source_Info** state the Policy Engine **Shall** request the present Source information from the Device Policy Manager and then send a **Source_Info** Message based on this information.

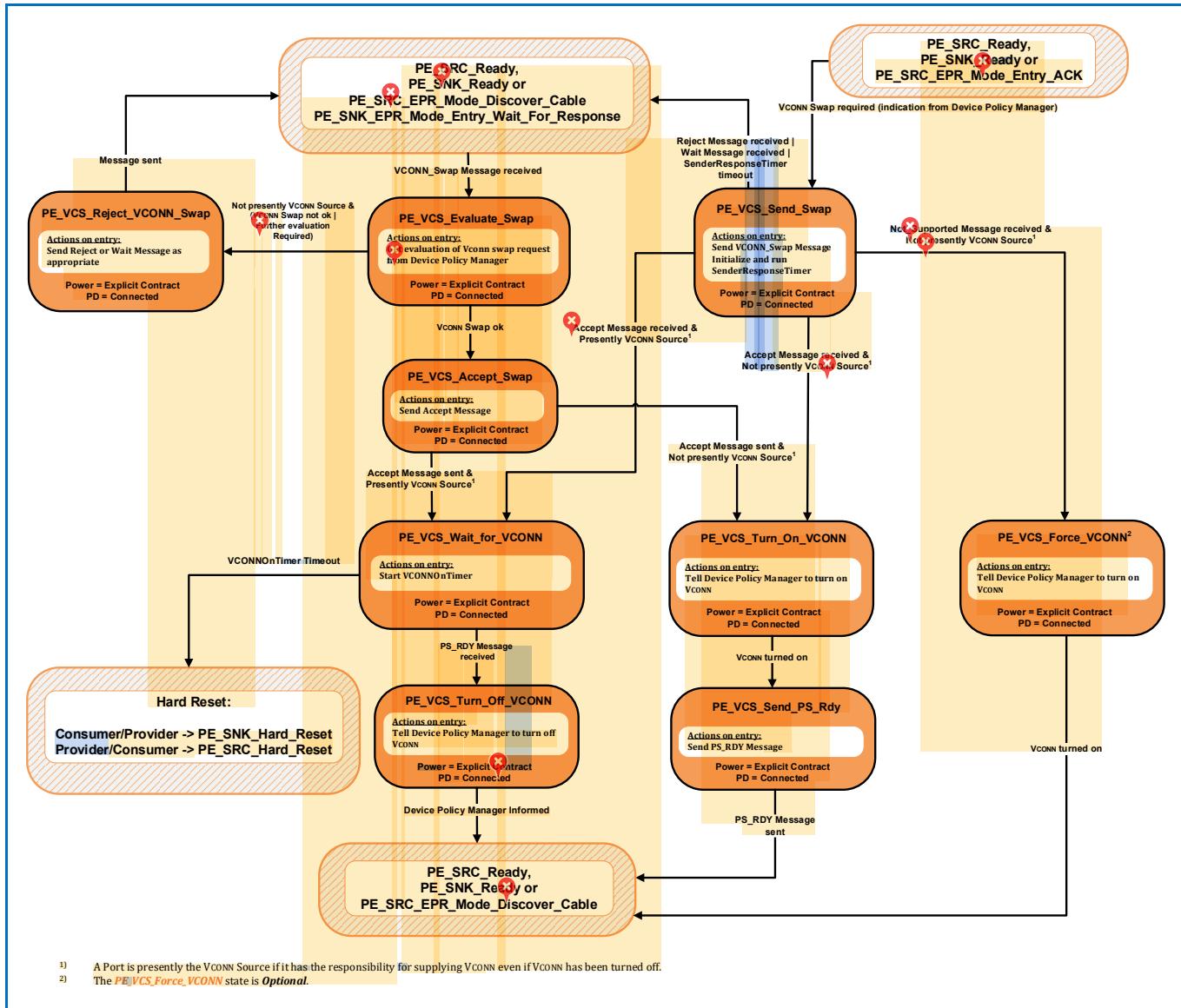
The Policy Engine **Shall** transition back to the **PE_SNK_Ready** state (see [Figure 8-135 “Sink Port State Diagram”](#)) when:

- The **Source_Info** Message has been successfully sent.

8.3.3.21 VCONN Swap State Diagram

The State Diagram in this section **Shall** apply to Ports that supply VCONN. [Figure 8-193 "Vconn Swap State Diagram"](#) shows the state operation for a Port on sending or receiving a VCONN Swap request.

[Figure 8-193 "Vconn Swap State Diagram"](#)



8.3.3.21.1 PE_VCS_Send_Swap State

The **PE_VCS_Send_Swap** state is entered from either the **PE_SRC_Ready** or **PE_SNK_Ready** state when the Policy Engine receives a request from the Device Policy Manager to perform a VCONN Swap.

On entry to the **PE_VCS_Send_Swap** state the Policy Engine **Shall** send a **VCONN_Swap Message** and start the **SenderResponseTimer**.

The Policy Engine **Shall** transition to the **PE_VCS_Wait_For_VCONN** state when:

- An **Accept** Message is received and **PE_VCS_Swap**

- The Port is presently the VCONN Source.

The Policy Engine ***Shall*** transition to the ***PE_VCS_Turn_On_VCONN*** state when:

- An ***Accept*** Message is received and
- The Port is not presently the VCONN Source.

The Policy Engine ***Shall*** transition back to either the ***PE_SRC_Ready***, ***PE_SNK_Ready*** or ***PE_SRC_EPR_Mode_Discover_Cable*** state when:

- A ***Reject*** Message is received or
- A ***Wait*** Message is received or
- The ***SenderResponseTimer*** times out.

The Policy Engine ***May*** transition to the ***PE_VCS_Force_VCONN*** state when:

- A ***Not_Supported*** Message is received and
- ~~☒~~The Port is not presently the VCONN Source.

8.3.3.21.2 PE_VCS_Evaluate_Swap State

The ***PE_VCS_Evaluate_Swap*** state is entered from either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when the Policy Engine receives a ***VCONN_Swap*** Message.

On entry to the ***PE_VCS_Evaluate_Swap*** state the Policy Engine ***Shall*** request the Device Policy Manager for an evaluation of the VCONN Swap request. Note Ports that are presently the VCONN Source must always accept a VCONN swap request (see [Section 6.3.11 "VCONN_Swap Message"](#)).

The Policy Engine ***Shall*** transition to the ***PE_VCS_Accept_Swap*** state when:

- The Device Policy Manager indicates that a VCONN Swap is ok.

The Policy Engine ***Shall*** transition to the ***PE_VCS_Reject_Swap*** state when:

- The Port is not presently the VCONN Source and
- The Device Policy Manager indicates that a VCONN Swap is not ok or
- The Device Policy Manager indicates that a VCONN Swap cannot be done at this time.

8.3.3.21.3 PE_VCS_Accept_Swap State

On entry to the ***PE_VCS_Accept_Swap*** state the Policy Engine ***Shall*** send an ***Accept*** Message.

The Policy Engine ***Shall*** transition to the ***PE_VCS_Wait_For_VCONN*** state when:

- The ***Accept*** Message has been sent and
- The Port's VCONN is on.

The Policy Engine ***Shall*** transition to the ***PE_VCS_Turn_On_VCONN*** state when:

- The ***Accept*** Message has been sent and
- The Port's VCONN is off.

8.3.3.21.4 PE_VCS_Reject_Swap State

On entry to the ***PE_VCS_Reject_Swap*** state the Policy Engine ***Shall*** request the Protocol Layer to send:

- A **Reject** Message if the device is unable to perform a VCONN Swap at this time.
- A **Wait** Message if further evaluation of the VCONN Swap request is required. Note in this case it is expected that the Port will send a **VCONN_Swap** Message at a later time.

The Policy Engine **Shall** transition back to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_SRC_EPR_Mode_Discover_Cable** state when:

- The **Reject** or **Wait** Message has been sent.

8.3.3.21.5 PE_VCS_UFP_Wait_for_VCONN State

On entry to the **PE_VCS_Wait_For_VCONN** state the Policy Engine **Shall** start the **VCONNOnTimer**.

The Policy Engine **Shall** transition to the **PE_VCS_Turn_Off_VCONN** state when:

- A **PS_RDY** Message is received.

The Policy Engine **Shall** transition to either the **PE_SRC_Hard_Reset** or **PE_SNK_Hard_Reset** state when:

- The **VCONNOnTimer** times out.

8.3.3.21.6 PE_VCS_Turn_Off_VCONN State

On entry to the **PE_VCS_Turn_Off_VCONN** state the Policy Engine **Shall** tell the Device Policy Manager to turn off VCONN.

The Policy Engine **Shall** transition back to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_SRC_EPR_Mode_Discover_Cable** state when:

- The Device Policy Manager has been informed.

8.3.3.21.7 PE_VCS_Turn_On_VCONN State

On entry to the **PE_VCS_Turn_On_VCONN** state the Policy Engine **Shall** tell the Device Policy Manager to turn on VCONN.

The Policy Engine **Shall** transition to the **PE_VCS_Send_Ps_Rdy** state when:

- The Port's VCONN is on.

8.3.3.21.8 PE_VCS_Send_PS_Rdy State

On entry to the **PE_VCS_Send_Ps_Rdy** state the Policy Engine **Shall** send a **PS_RDY** Message.

The Policy Engine **Shall** transition back to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_SRC_EPR_Mode_Discover_Cable** state when:

- The **PS_RDY** Message has been sent.

8.3.3.21.9 PE_VCS_Force_VCONN State

On entry to the **PE_VCS_Force_VCONN** state the Policy Engine **Shall** tell the Device Policy Manager to turn on VCONN.

The Policy Engine **Shall** transition back to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_SRC_EPR_Mode_Discover_Cable** state when:

- The Port's VCONN is on.

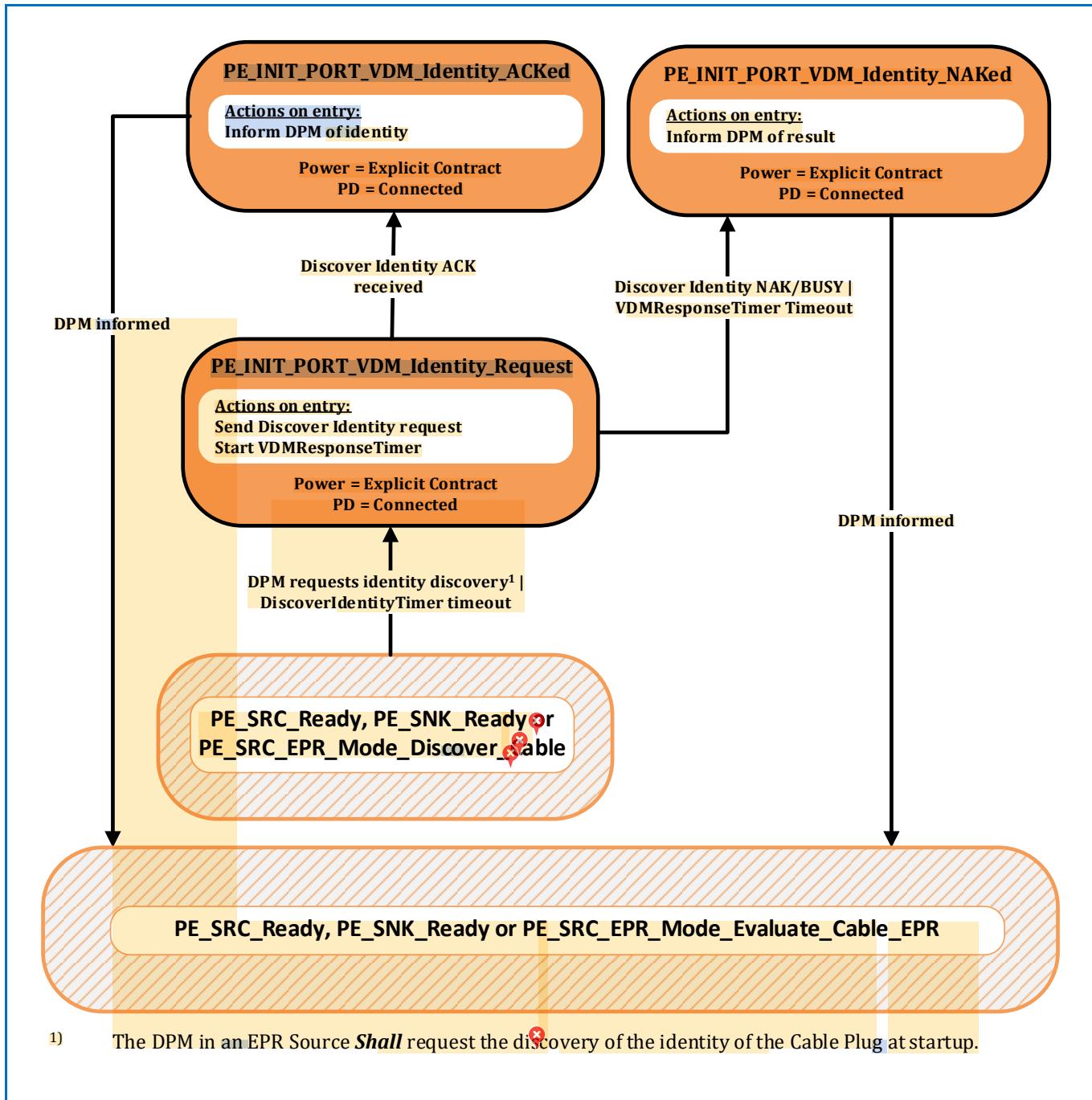
8.3.3.22 Initiator Structured VDM State Diagrams

 The State Diagrams in this section *Shall* apply to all Initiators.

8.3.3.22.1 Initiator Structured VDM Discover Identity State Diagram

Figure 8-194 “Initiator to Port VDM Discover Identity State Diagram” shows the state diagram for an Initiator when discovering the identity of its Port Partner or Cable Plug.

Figure 8-194 “Initiator to Port VDM Discover Identity State Diagram”



8.3.3.22.1.1 PE_INIT_PORT_VDM_Identity_Request State

The Policy Engine transitions to the **PE_INIT_PORT_VDM_Identity_Request** state from either the **PE_SRC_Ready** or **PE_SNK_Ready** state when:

- The Device Policy Manager requests the discovery of the identity of the Port Partner or Cable Plug or
- The **DiscoverIdentityTimer** times out.

The Policy Engine transitions to the **PE_INIT_PORT_VDM_Identity_Request** state from the **PE_SRC_EPR_Mode_Discover_Cable** state when:

- The Cable Plug discovery process has been initiated.

On entry to the **PE_INIT_PORT_VDM_Identity_Request** state the Policy Engine **Shall** send a Structured VDM **Discover Identity** Command request and **Shall** start the **VDMResponseTimer**.

The Policy Engine **Shall** transition to the **PE_INIT_PORT_VDM_Identity_ACKed** state when:

- A Structured VDM **Discover Identity** ACK Command response is received.

The Policy Engine **Shall** transition to the **PE_INIT_PORT_VDM_Identity_NAKed** state when:

- A Structured VDM **Discover Identity** NAK or BUSY Command response is received or
- The **VDMResponseTimer** times out.

8.3.3.22.1.2 PE_INIT_PORT_VDM_Identity_ACKed State

On entry to the **PE_INIT_PORT_VDM_Identity_ACKed** state the Policy Engine **Shall** inform the Device Policy Manager of the Identity information.

The Policy Engine **Shall** transition to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_SRC_EPR_Mode_Evaluate_Cable_EPR** state when:

- The Device Policy Manager has been informed.

8.3.3.22.1.3 PE_INIT_PORT_VDM_Identity_NAKed State

On entry to the **PE_INIT_PORT_VDM_Identity_NAKed** state the Policy Engine **Shall** inform the Device Policy Manager of the result (NAK, BUSY or timeout).

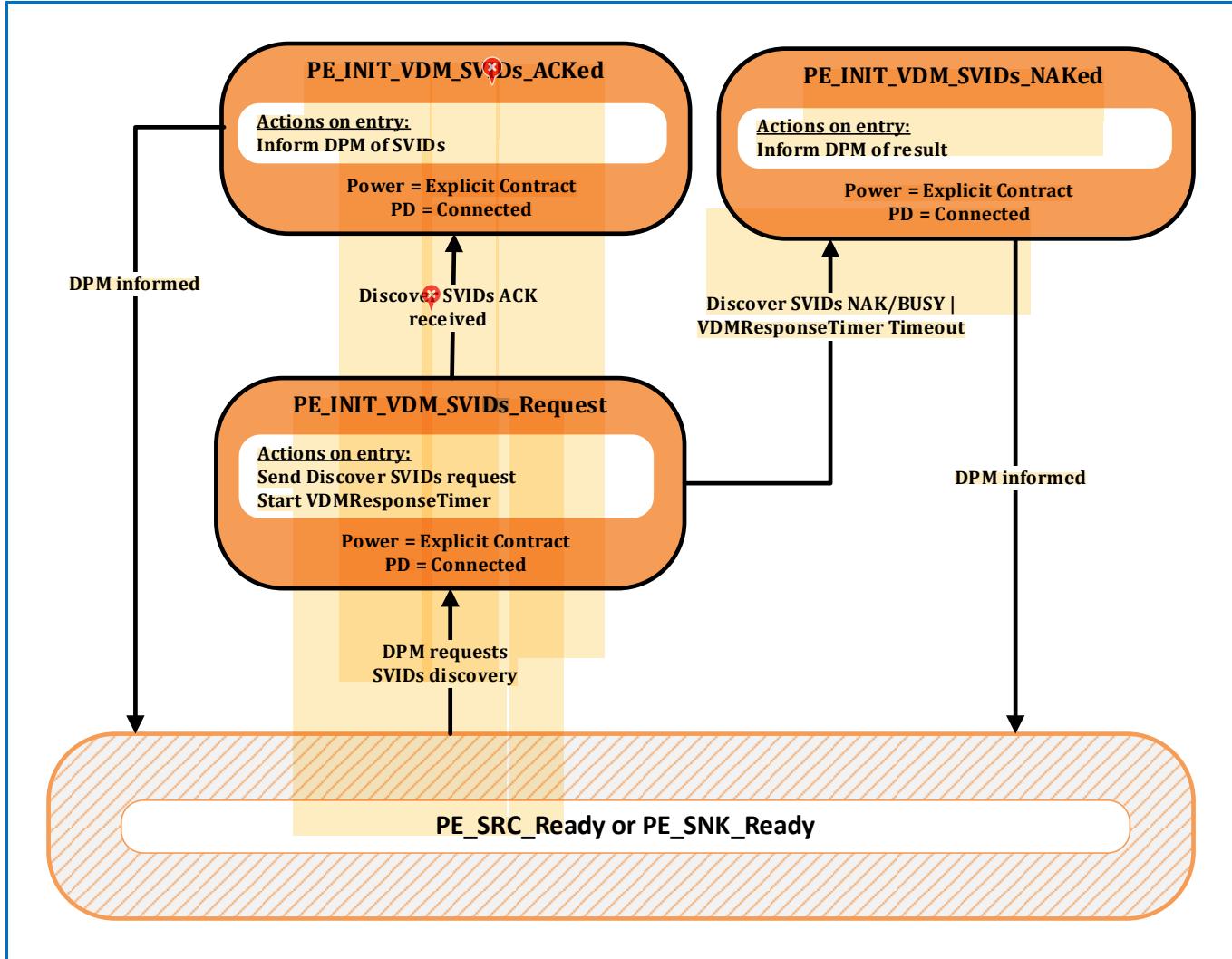
The Policy Engine **Shall** transition to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_SRC_EPR_Mode_Evaluate_Cable_EPR** state when:

- The Device Policy Manager has been informed.

8.3.3.22.2 Initiator Structured VDM Discover SVIDs State Diagram

Figure 8-195 “Initiator VDM Discover SVIDs State Diagram” shows the state diagram for an Initiator when discovering SVIDs of its Port Partner or Cable Plug.

Figure 8-195 “Initiator VDM Discover SVIDs State Diagram”



8.3.3.22.2.1 PE_INIT_VDM_SVIDs_Request State

The Policy Engine transitions to the **PE_INIT_VDM_SVIDs_Request** state from either the **PE_SRC_Ready** or **PE_SNK_Ready** state when:

- The Device Policy Manager requests the discovery of the SVIDs of the Port Partner or a Cable Plug.

On entry to the **PE_INIT_VDM_SVIDs_Request** state the Policy Engine **Shall** send a Structured VDM **Discover SVIDs** Command request and **Shall** start the **VDMResponseTimer**.

The Policy Engine **Shall** transition to the **PE_INIT_VDM_SVIDs_ACKed** state when:

- A Structured VDM **Discover SVIDs** ACK Command response is received.

The Policy Engine **Shall** transition to the **PE_INIT_VDM_SVIDs_NAKed** state when:

- A Structured VDM ***Discover SVIDs*** NAK or BUSY Command response is received or
- The ***VDMResponseTimer*** times out.

8.3.3.22.2.2 PE_INIT_VDM_SVIDs_ACKed State

On entry to the ***PE_INIT_VDM_SVIDs_ACKed*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the SVIDs information.

The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when:

- The Device Policy Manager has been informed.

8.3.3.22.2.3 PE_INIT_VDM_SVIDs_NAKed State

On entry to the ***PE_INIT_VDM_SVIDs_NAKed*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the result (NAK, BUSY or timeout).

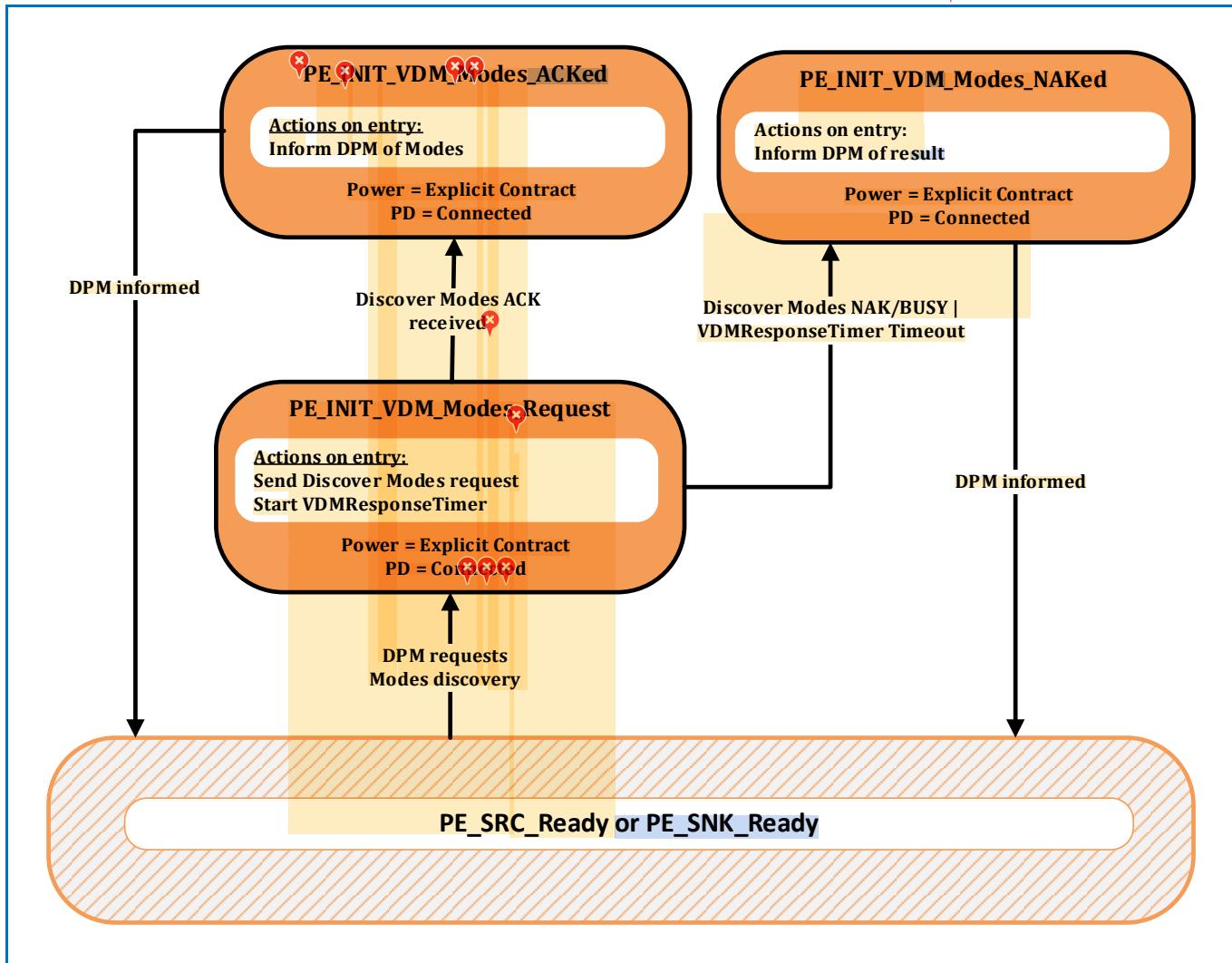
The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state when:

- The Device Policy Manager has been informed.

8.3.3.22.3 Initiator Structured VDM Discover Modes State Diagram

Figure 8-196 “Initiator VDM Discover Modes State Diagram” shows the state diagram for an Initiator when discovering Modes of its Port Partner or Cable Plug.

Figure 8-196 “Initiator VDM Discover Modes State Diagram”



8.3.3.22.3.1 PE_INIT_VDM_Modes_Request State

The Policy Engine transitions to the **PE_INIT_VDM_Modes_Request** state from either the **PE_SRC_Ready** or **PE_SNK_Ready** state when:

- The Device Policy Manager requests the discovery of the Modes of the Port Partner or a Cable Plug.

On entry to the **PE_INIT_VDM_Modes_Request** state the Policy Engine **Shall** send a Structured VDM **Discover Modes** Command request and **Shall** start the **VDMResponseTimer**.

The Policy Engine **Shall** transition to the **PE_INIT_VDM_Modes_ACKed** state when:

- A Structured VDM **Discover Modes** ACK Command response is received.

The Policy Engine **Shall** transition to the **PE_INIT_VDM_Modes_NAKed** state when:

- A Structured VDM ***Discover Modes*** NAK or BUSY Command response is received or
- ✖ • The ***VDMResponseTimer*** times out.

8.3.3.22.3.2 PE_INIT_VDM_Modes_ACKed State

On entry to the ***PE_INIT_VDM_Modes_ACKed*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the Modes information.

The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state for a DFP when:

- The Device Policy Manager has been informed.

8.3.3.22.3.3 PE_INIT_VDM_Modes_NAKed State

On entry to the ***PE_INIT_VDM_Modes_NAKed*** state the Policy Engine ***Shall*** inform the Device Policy Manager of the result (NAK, BUSY or timeout).

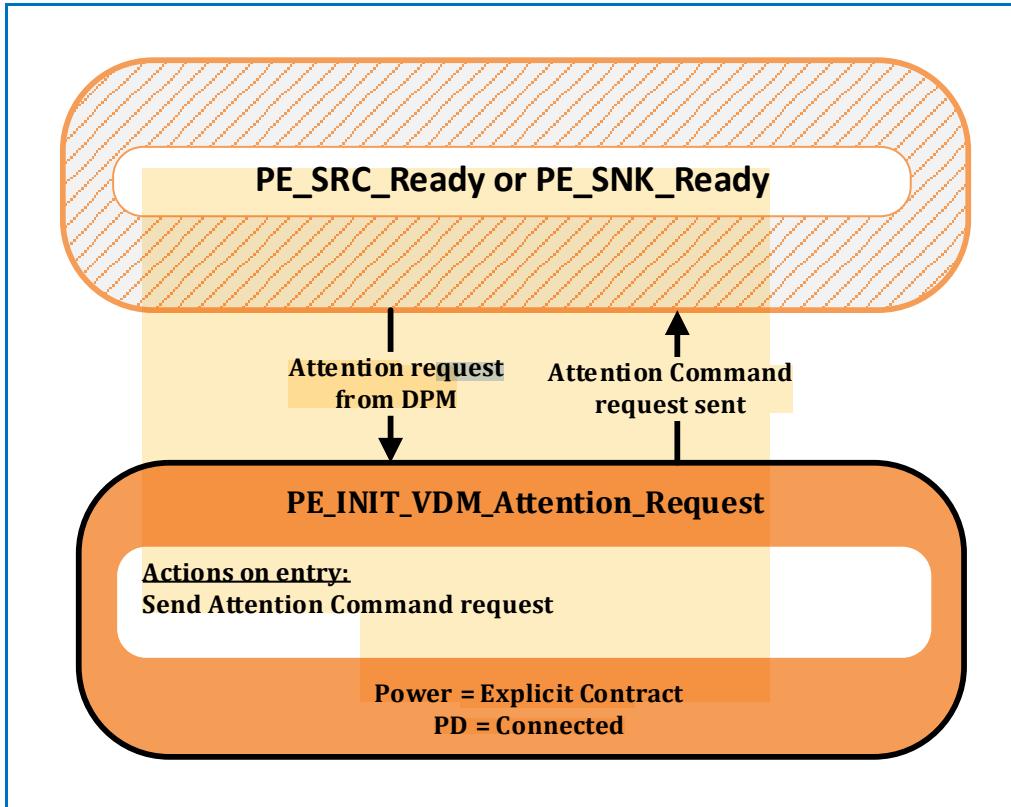
The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state for a DFP when:

- The Device Policy Manager has been informed.

8.3.3.22.4 Initiator Structured VDM Attention State Diagram

Figure 8-197 “Initiator VDM Attention State Diagram” shows the state diagram for an Initiator when sending an **Attention** Command request.

Figure 8-197 “Initiator VDM Attention State Diagram”



8.3.3.22.4.1 PE_INIT_VDM_Attention_Request State

The Policy Engine transitions to the **PE_INIT_VDM_Attention_Request** state from either the **PE_SRC_Ready** or **PE_SNK_Ready** state when:

- When the Device Policy Manager requests attention from its Port Partner.

On entry to the **PE_INIT_VDM_Attention_Request** state the Policy Engine **Shall** send an **Attention** Command request.

The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state when:

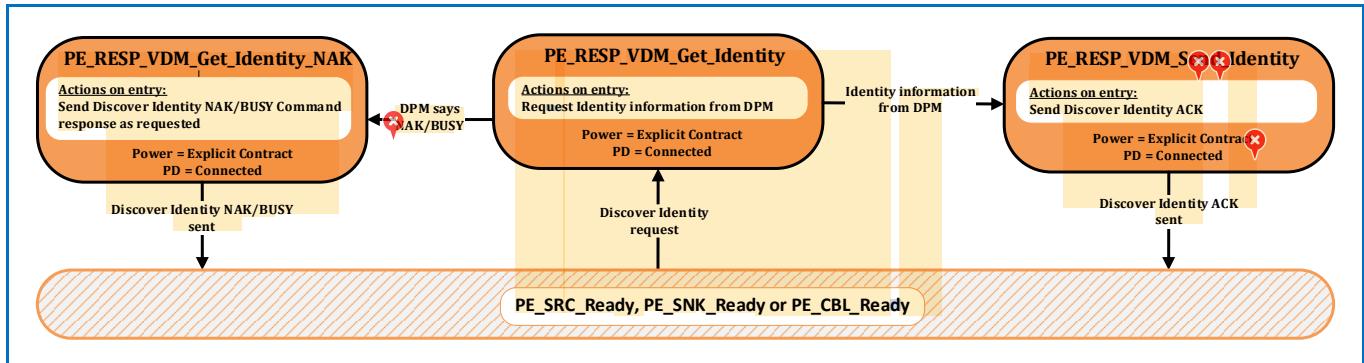
- The **Attention** Command request has been sent.

8.3.3.23 Responder Structured VDM State Diagrams

8.3.3.23.1 Responder Structured VDM Discover Identity State Diagram

Figure 8-198 “Responder Structured VDM Discover Identity State Diagram” shows the state diagram for a Responder receiving a *Discover Identity* Command request.

Figure 8-198 “Responder Structured VDM Discover Identity State Diagram”



8.3.3.23.1.1 PE_RESP_VDM_Get_Identity State

The Policy Engine transitions to the **PE_RESP_VDM_Get_Identity** state from either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state when:

- A Structured VDM *Discover Identity* Command request is received.

On entry to the **PE_RESP_VDM_Get_Identity** state the Responder **Shall** request identity information from the Device Policy Manager.

The Policy Engine **Shall** transition to the **PE_RESP_VDM_Send_Identity** state when:

- Identity information is received from the Device Policy Manager.

The Policy Engine **Shall** transition to the **PE_RESP_VDM_Get_Identity_NAK** state when:

- The Device Policy Manager indicates that the response to the *Discover Identity* Command request is NAK or BUSY.

8.3.3.23.1.2 PE_RESP_VDM_Send_Identity State

On entry to the **PE_RESP_VDM_Send_Identity** state the Responder **Shall** send the Structured VDM *Discover Identity* ACK Command response.

The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state for a UFP when:

- The Structured VDM *Discover Identity* ACK Command response has been sent.

8.3.3.23.1.3 PE_RESP_VDM_Get_Identity_NAK State

On entry to the **PE_RESP_VDM_Get_Identity_NAK** state the Policy Engine **Shall** send a Structured VDM *Discover Identity* NAK or BUSY Command response as indicated by the Device Policy Manager.

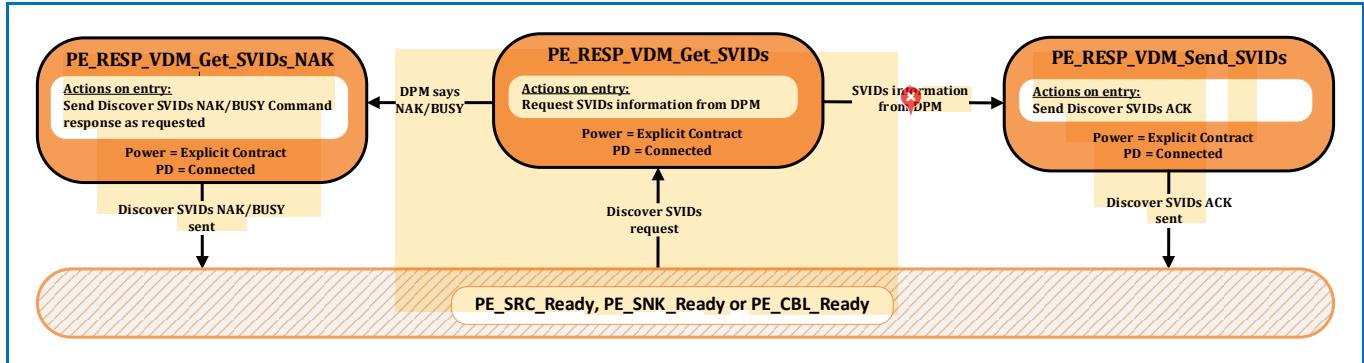
The Policy Engine **Shall** transition to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state when:

- The Structured VDM *Discover Identity* NAK or BUSY Command response has been sent.

8.3.3.23.2 Responder Structured VDM Discover SVIDs State Diagram

Figure 8-199 “Responder Structured VDM Discover SVIDs State Diagram” shows the state diagram for a Responder when receiving a **Discover SVIDs** Command.

Figure 8-199 “Responder Structured VDM Discover SVIDs State Diagram”



8.3.3.23.2.1 PE_RESP_VDM_Get_SVIDs State

The Policy Engine transitions to the **PE_RESP_VDM_Get_SVIDs** state from either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state when:

- A Structured VDM **Discover SVIDs** Command request is received.

On entry to the **PE_RESP_VDM_Get_SVIDs** state the Responder **Shall** request SVIDs information from the Device Policy Manager.

The Policy Engine **Shall** transition to the **PE_RESP_VDM_Send_SVIDs** state when:

- SVIDs information is received from the Device Policy Manager.

The Policy Engine **Shall** transition to the **PE_RESP_VDM_Get_SVIDs_NAK** state when:

- The Device Policy Manager indicates that the response to the **Discover SVIDs** Command request is NAK or BUSY.

8.3.3.23.2.2 PE_UFP_VDM_Send_SVIDs State

On entry to the **PE_RESP_VDM_Send_SVIDs** state the Responder **Shall** send the Structured VDM **Discover SVIDs** ACK Command response.

The Policy Engine **Shall** transition to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state when:

- The Structured VDM **Discover SVIDs** ACK Command response has been sent.

8.3.3.23.2.3 PE_UFP_VDM_Get_SVIDs_NAK State

On entry to the **PE_RESP_VDM_Get_SVIDs_NAK** state the Policy Engine **Shall** send a Structured VDM **Discover SVIDs** NAK or BUSY Command response as indicated by the Device Policy Manager.

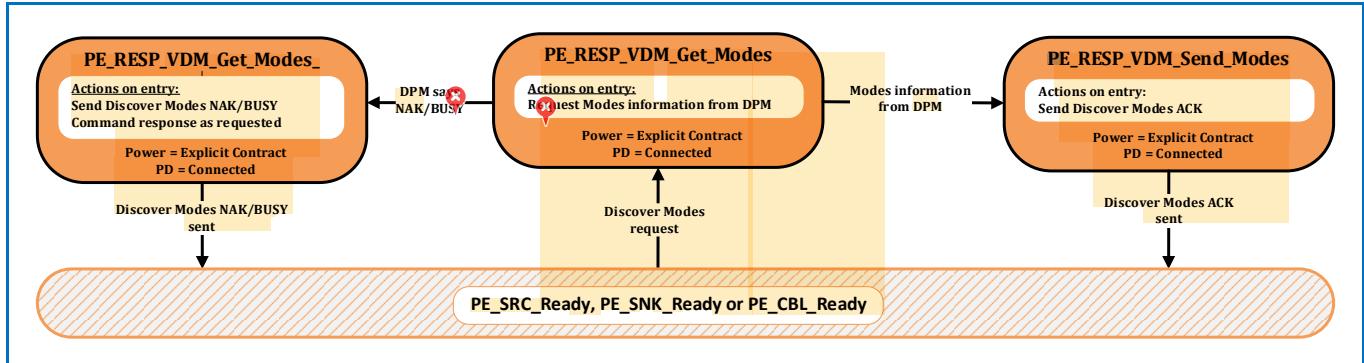
The Policy Engine **Shall** transition to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state when:

- The Structured VDM **Discover SVIDs** NAK or BUSY Command response has been sent.

8.3.3.23.3 Responder Structured VDM Discover Modes State Diagram

Figure 8-200 “Responder Structured VDM Discover Modes State Diagram” shows the state diagram for a Responder on receiving a **Discover Modes** Command.

Figure 8-200 “Responder Structured VDM Discover Modes State Diagram”



8.3.3.23.3.1 PE_RESP_VDM_Get_Modes State

The Policy Engine transitions to the **PE_RESP_VDM_Get_Modes** state from either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state when:

- A Structured VDM **Discover Modes** Command request is received.

On entry to the **PE_RESP_VDM_Get_Modes** state the Responder **Shall** request Modes information from the Device Policy Manager.

The Policy Engine **Shall** transition to the **PE_RESP_VDM_Send_Modes** state when:

- Modes information is received from the Device Policy Manager.

The Policy Engine **Shall** transition to the **PE_RESP_VDM_Get_Modes_NAK** state when:

- The Device Policy Manager indicates that the response to the **Discover Modes** Command request is NAK or BUSY.

8.3.3.23.3.2 PE_RESP_VDM_Send_Modes State

On entry to the **PE_RESP_VDM_Send_Modes** state the Responder **Shall** send the Structured VDM **Discover Modes** ACK Command response.

The Policy Engine **Shall** transition to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state when:

- The Structured VDM **Discover Modes** ACK Command response has been sent.

8.3.3.23.3.3 PE_RESP_VDM_Get_Modes_NAK State

On entry to the **PE_RESP_VDM_Get_Modes_NAK** state the Policy Engine **Shall** send a Structured VDM **Discover Modes** NAK or BUSY Command response as indicated by the Device Policy Manager.

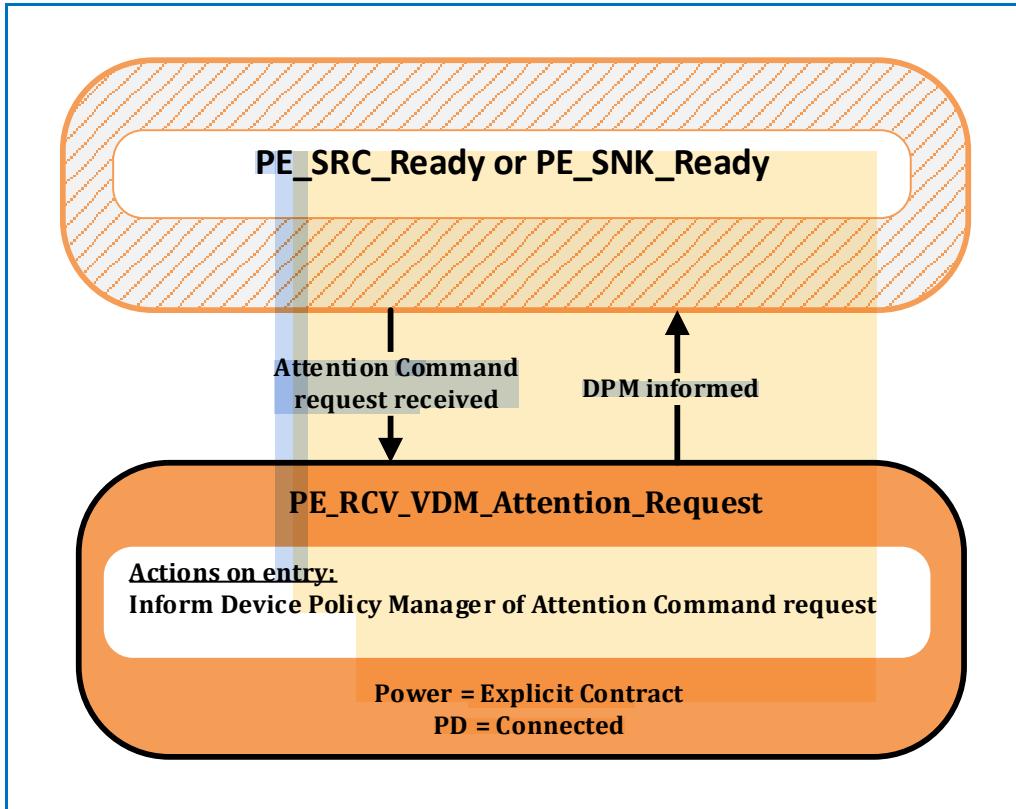
The Policy Engine **Shall** transition to either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state when:

- The Structured VDM **Discover Modes** NAK or BUSY Command response has been sent.

8.3.3.23.4 Receiving a Structured VDM Attention State Diagram

Figure 8-201 “Receiving a Structured VDM Attention State Diagram” shows the state diagram when receiving an **Attention** Command request.

Figure 8-201 “Receiving a Structured VDM Attention State Diagram”



8.3.3.23.4.1 PE_RCV_VDM_Attention_Request State

The Policy Engine transitions to the **PE_RCV_VDM_Attention_Request** state from either the **PE_SRC_Ready** or **PE_SNK_Ready** state when:

- An **Attention** Command request is received.

On entry to the **PE_RCV_VDM_Attention_Request** state the Policy Engine **Shall** inform the Device Policy Manager of the **Attention** Command request.

The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state when:

- The Device Policy Manager has been informed.

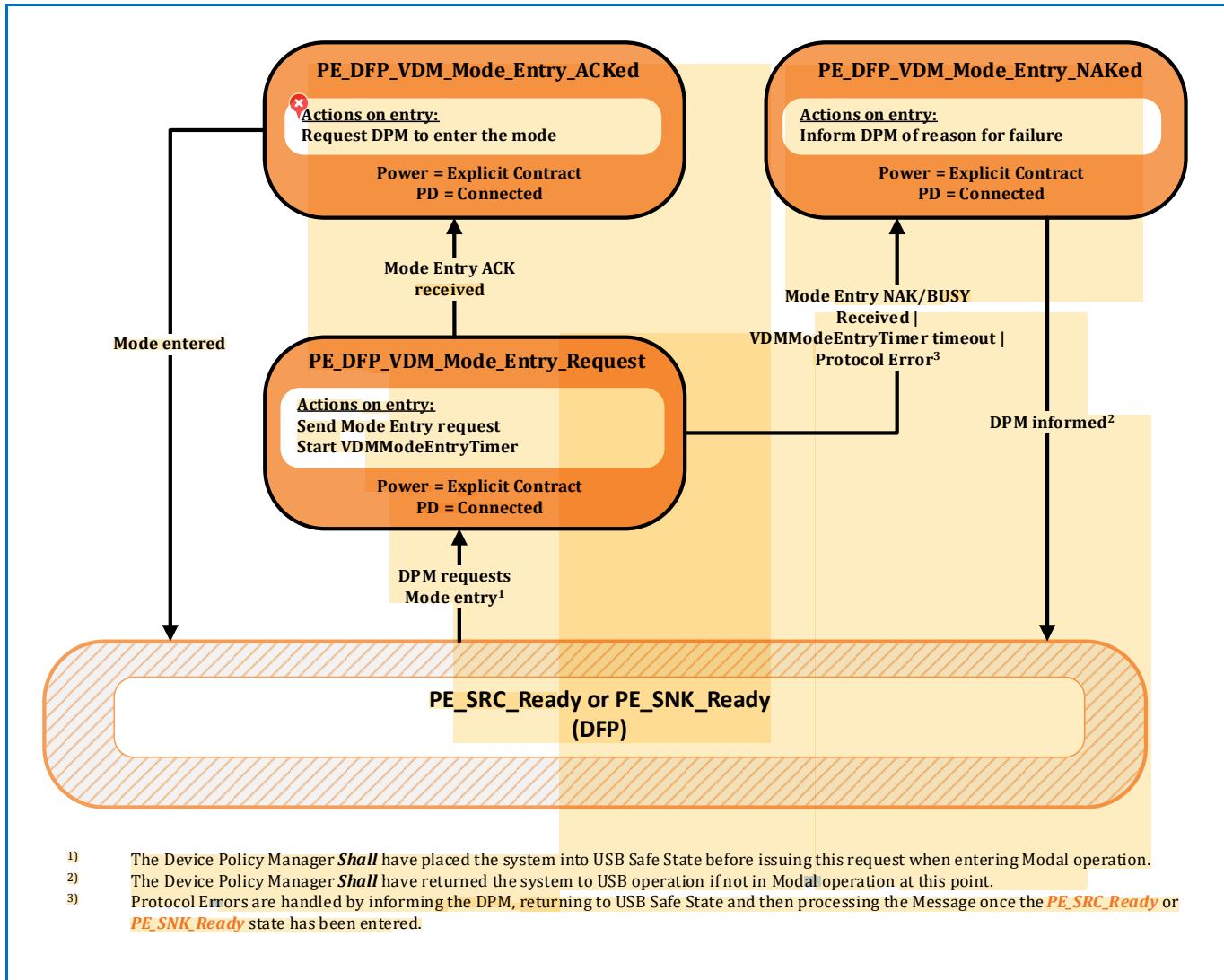
8.3.3.24 DFP Structured VDM State Diagrams

The State Diagrams in this section **Shall** apply to all DFPs that support Structured VDMs.

8.3.3.24.1 DFP Structured VDM Mode Entry State Diagram

Figure 8-202 "DFP VDM Mode Entry State Diagram" shows the state operation for a DFP when entering a Mode.

Figure 8-202 "DFP VDM Mode Entry State Diagram"



8.3.3.24.1.1 PE_DFP_VDM_Mode_Entry_Request State

The Policy Engine transitions to the **PE_DFP_VDM_Mode_Entry_Request** state from either the **PE_SRC_Ready** or **PE_SNK_Ready** state for a DFP when:

- The Device Policy Manager requests that the Port Partner or a Cable Plug enter a Mode.

On entry to the **PE_DFP_VDM_Mode_Entry_Request** state the Policy Engine **Shall** send a Structured VDM **Enter Mode** Command request and **Shall** start the **VDMModeEntryTimer**.

The Policy Engine **Shall** transition to the **PE_DFP_VDM_Mode_Entry_ACKed** state when:

- A Structured VDM **Enter Mode** ACK Command response is received.

The Policy Engine **Shall** transition to the **PE_DFP_VDM_Mode_Entry_NAKed** state when:

- A Structured VDM **Enter Mode** NAK or BUSY Command response is received or
- The **VDMModeEntryTimer** times out.

8.3.3.24.1.2 PE_DFP_VDM_Mode_Entry_ACKed State

On entry to the **PE_DFP_VDM_Mode_Entry_ACKed** state the Policy Engine **Shall** request the Device Policy Manager to enter the Mode.

The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state for a DFP when:

- The Mode has been entered.

8.3.3.24.1.3 PE_DFP_VDM_Mode_Entry_NAKed State

On entry to the **PE_DFP_VDM_Mode_Entry_NAKed** state the Policy Engine **Shall** inform the Device Policy Manager of the reason for failure (NAK, BUSY, timeout or Protocol Error).

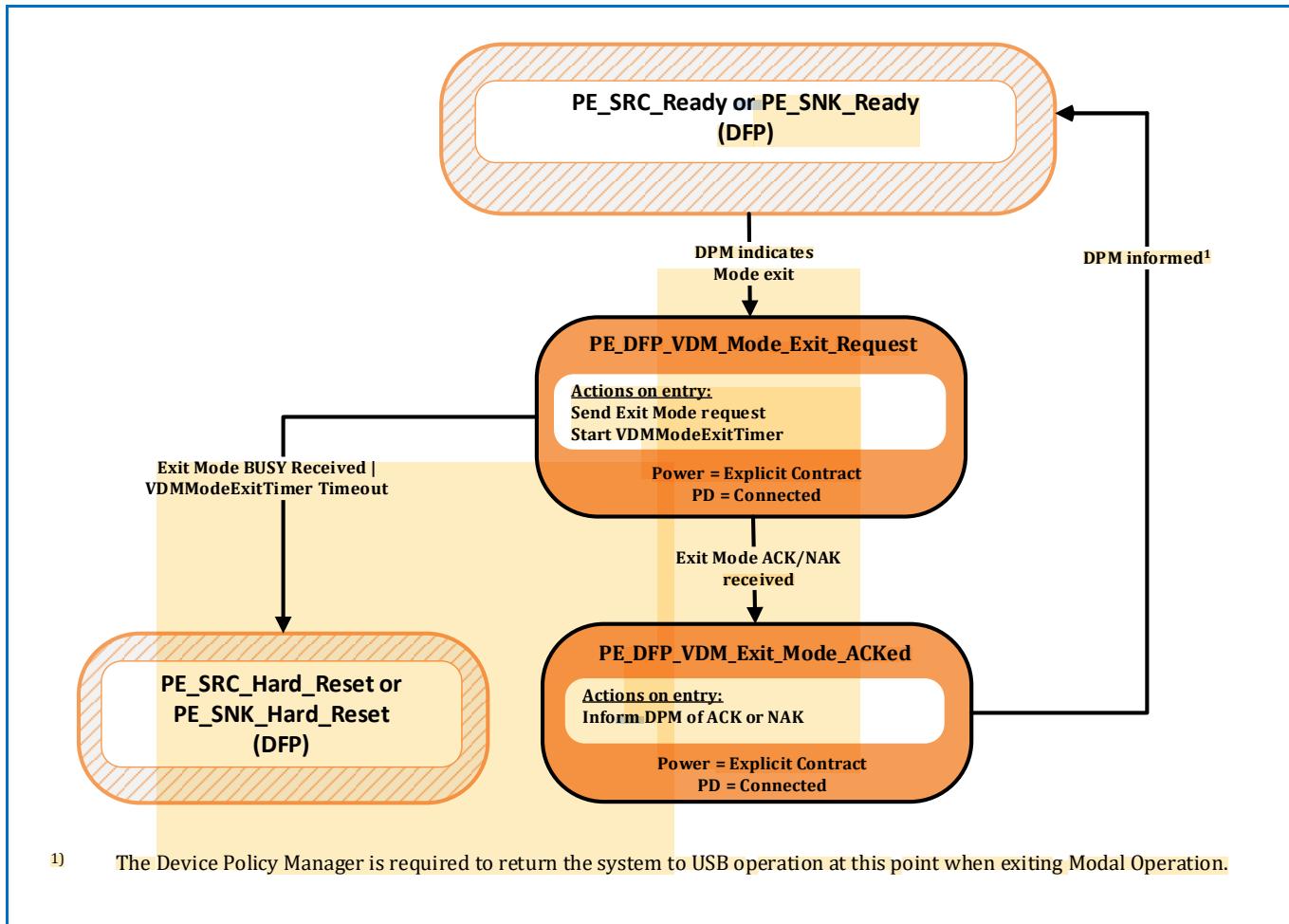
The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state for a DFP when:

- The Device Policy Manager has been informed.

8.3.3.24.2 DFP Structured VDM Mode Exit State Diagram

Figure 8-203 “DFP VDM Mode Exit State Diagram” shows the state diagram for a DFP when exiting a Mode.

Figure 8-203 “DFP VDM Mode Exit State Diagram”



8.3.3.24.2.1 PE_DFP_VDM_Mode_Exit_Request State

The Policy Engine transitions to the **PE_DFP_VDM_Mode_Exit_Request** state from either the **PE_SRC_Ready** or **PE_SNK_Ready** state for a DFP when:

- The Device Policy Manager requests that the Port Partner or a Cable Plug exit a Mode.

On entry to the **PE_DFP_VDM_Mode_Exit_Request** state the Policy Engine **Shall** send a Structured VDM **Exit Mode** Command request and **Shall** start the **VDMModeExitTimer**.

The Policy Engine **Shall** transition to the **PE_DFP_VDM_Mode_Exit_ACKed** state when:

- A Structured VDM **Exit Mode** ACK or NAK Command response is received.

The Policy Engine **Shall** transition to either the **PE_SRC_Hard_Reset** or **PE_SNK_Hard_Reset** state depending on the present Power Role when:

- A Structured VDM **Exit Mode** BUSY Command response is received or

- The **VDMModeExitTimer** times out.

8.3.3.24.2.2 PE_DFP_VDM_DFP_Mode_Exit_ACKed State

✖ On Exit to the **PE_DFP_VDM_Mode_Exit_ACKed** state the Policy Engine **Shall** inform the Device Policy Manager Of the result: ACK or NAK.

The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state for a DFP when:

- The Device Policy Manager has been informed.

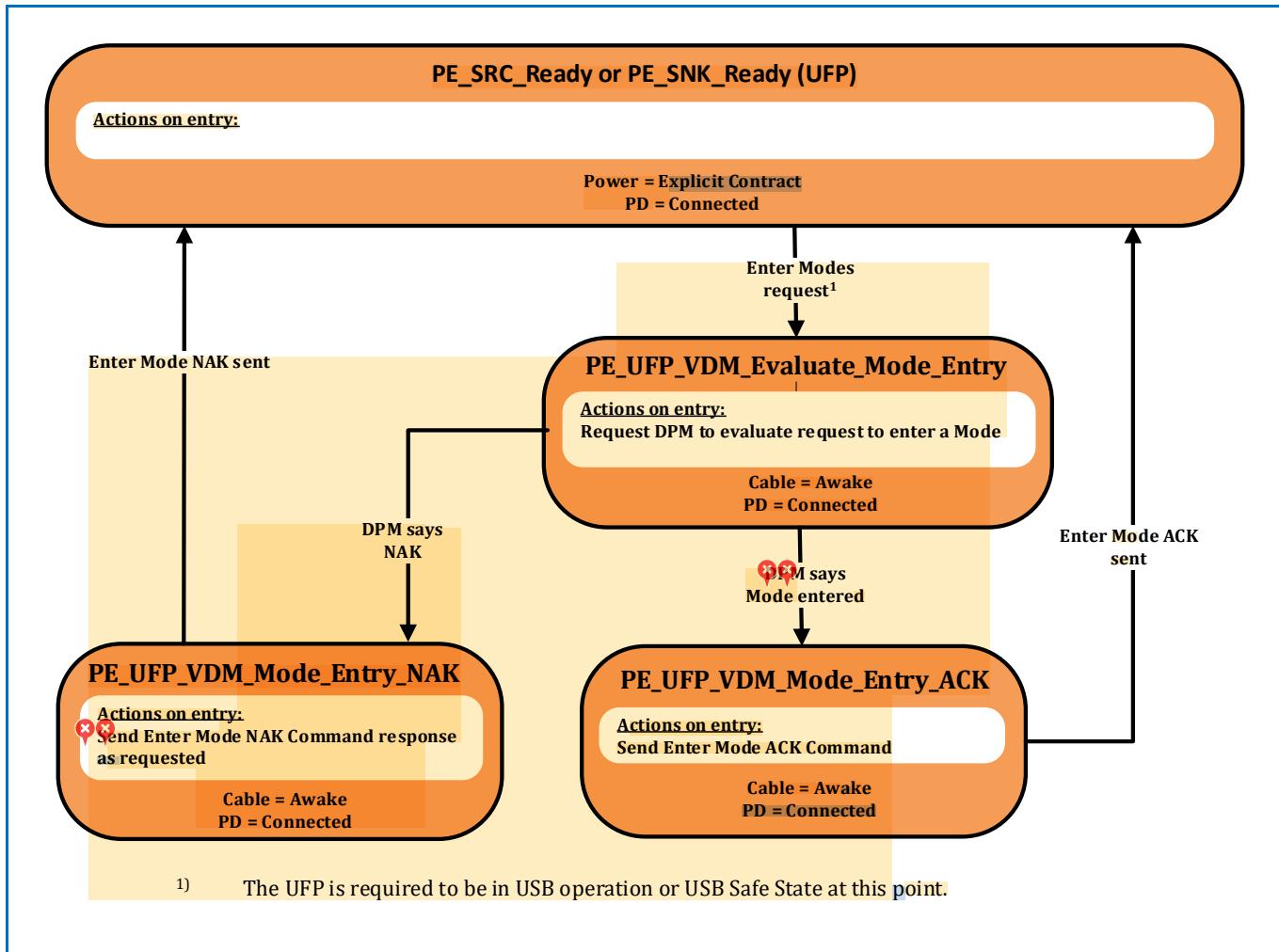
8.3.3.25 UFP Structured VDM State Diagrams

 The State Diagrams in this section **Shall** apply to all UFPs that support Structured VDMs.

8.3.3.25.1 UFP Structured VDM Enter Mode State Diagram

Figure 8-204 "UFP Structured VDM Enter Mode State Diagram" shows the state diagram for a UFP in response to an **Enter Mode** Command.

Figure 8-204 "UFP Structured VDM Enter Mode State Diagram"



8.3.3.25.1.1 PE_UFP_VDM_Evaluate_Mode_Entry State

The Policy Engine transitions to the **PE_UFP_VDM_Evaluate_Mode_Entry** state from either the **PE_SRC_Ready** or **PE_SNK_Ready** state for a UFP when:

- A Structured VDM **Enter Mode** Command request is received from the DFP.

On Entry to the **PE_UFP_VDM_Evaluate_Mode_Entry** state the Policy Engine **Shall** request the Device Policy Manager to evaluate the **Enter Mode** Command request and enter the Mode indicated in the Command request if the request is acceptable.

The Policy Engine **Shall** transition to the **PE_UFP_VDM_Mode_Entry_ACK** state when:

- The Device Policy Manager indicates that the Mode has been entered.

The Policy Engine ***Shall*** transition to the ***PE_UFP_VDM_Mode_Entry_NAK*** state when:

- The Device Policy Manager indicates that the response to the Mode request is NAK.

8.3.3.25.1.2 PE_UFP_VDM_Mode_Entry_ACK State

On entry to the ***PE_UFP_VDM_Mode_Entry_ACK*** state the Policy Engine ***Shall*** send a Structured VDM ***Enter Mode*** ACK Command response.

The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state for a UFP when:

- The Structured VDM ***Enter Mode*** ACK Command response has been sent.

8.3.3.25.1.3 PE_UFP_VDM_Mode_Entry_NAK State

On entry to the ***PE_UFP_VDM_Mode_Entry_NAK*** state the Policy Engine ***Shall*** send a Structured VDM ***Enter Mode*** NAK Command response as indicated by the Device Policy Manager.

The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state for a UFP when:

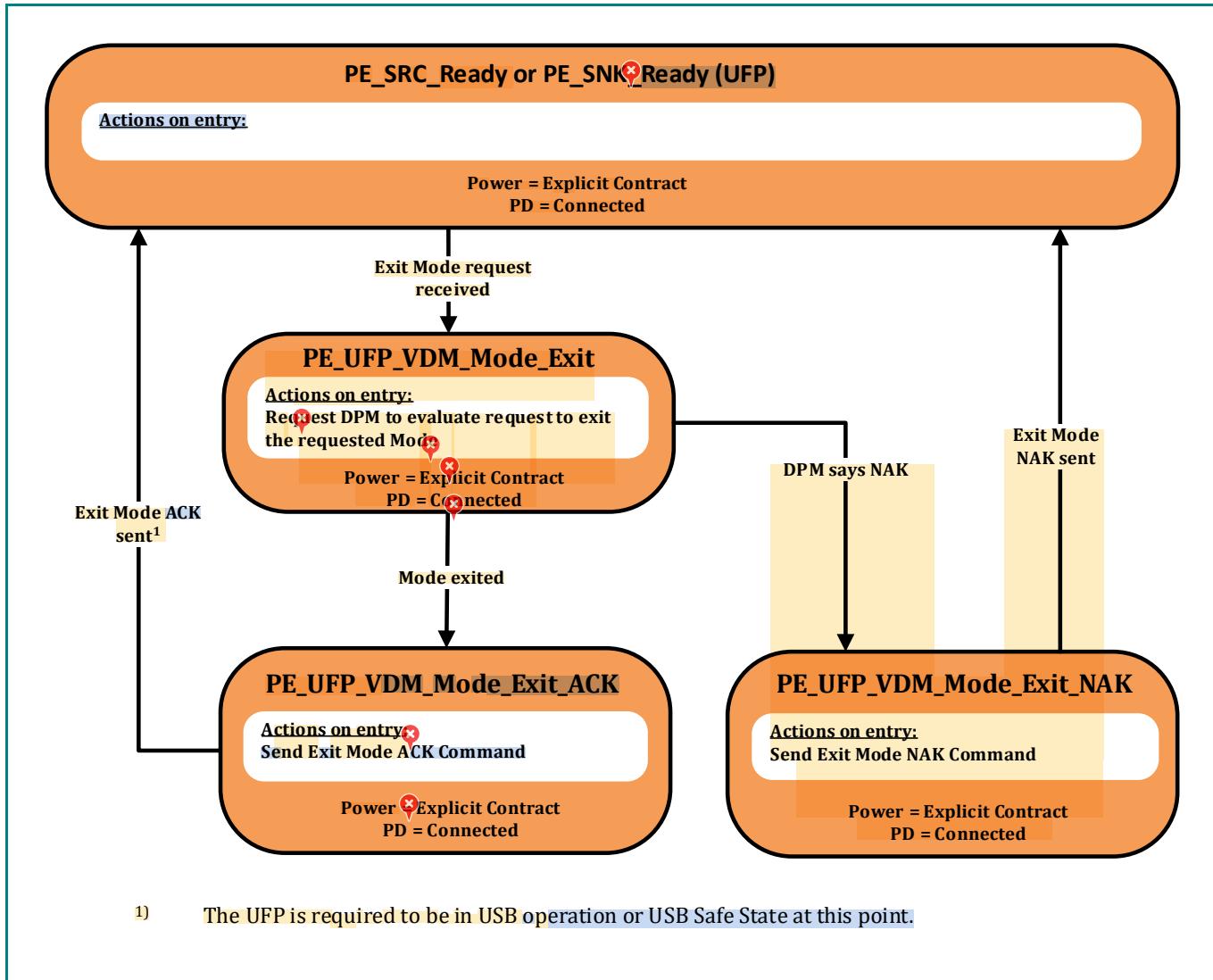
- The Structured VDM ***Enter Mode*** NAK Command response has been sent.

8.3.3.25.2

UFP Structured VDM Exit Mode State Diagram

Figure 8-205 “UFP Structured VDM Exit Mode State Diagram” shows the state diagram for a UFP in response to an **Exit Mode** Command.

Figure 8-205 “UFP Structured VDM Exit Mode State Diagram”



8.3.3.25.2.1

PE_UFP_VDM_Mode_Exit State

The Policy Engine transitions to the **PE_UFP_VDM_Mode_Exit** state from either the **PE_SRC_Ready** or **PE_SNK_Ready** state for a UFP when:

- A Structured VDM **Exit Mode** Command request is received from the DFP.

On entry to the **PE_UFP_VDM_Mode_Exit** state the Policy Engine **Shall** request the Device Policy Manager to exit the Mode indicated in the Command.

The Policy Engine **Shall** transition to the **PE_UFP_VDM_Mode_Exit_ACK** state when:

- The Device Policy Manager indicates that the Mode has been exited.

📍 The Policy Engine ***Shall*** transition to the ***PE_UFP_VDM_Mode_Exit_NAK*** state when:

- The Device Policy Manager indicates that the Command response to the ***Exit Mode*** Command request is NAK.

8.3.3.25.2.2 PE_CBL_Mode_Exit_ACK State

On entry to the ***PE_UFP_VDM_Mode_Exit_ACK*** state the Policy Engine ***Shall*** send a Structured VDM ***Exit Mode*** ACK Command response.

The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state for a UFP when:

- The Structured VDM ***Exit Mode*** ACK Command response has been sent.

8.3.3.25.2.3 PE_UFP_VDM_Mode_Exit_NAK State

On entry to the ***PE_UFP_VDM_Mode_Exit_NAK*** state the Policy Engine ***Shall*** send a Structured VDM ***Exit Mode*** NAK Command response as indicated by the Device Policy Manager.

The Policy Engine ***Shall*** transition to either the either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state for a UFP when:

- The Structured VDM ***Exit Mode*** NAK Command response has been sent.

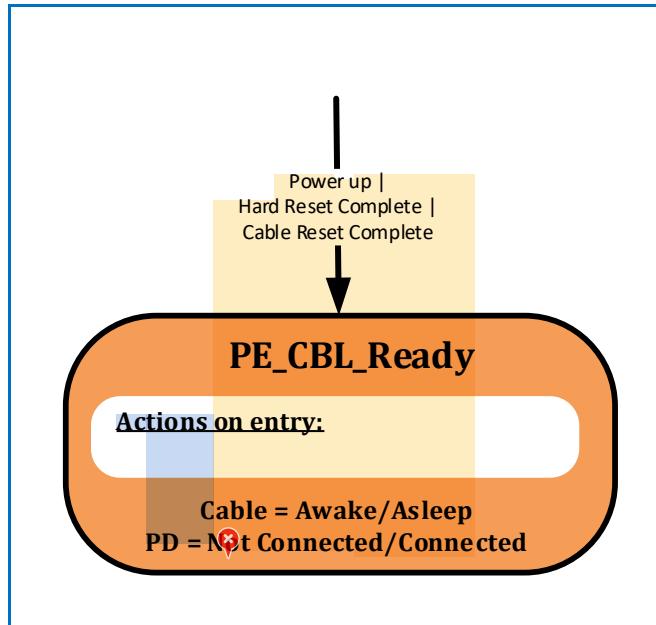
8.3.3.26 Cable Plug Specific State Diagrams

✖ The State Diagrams in this section **Shall** apply to all Cable Plugs that support Structured VDMs.

8.3.3.26.1 Cable Plug Cable Ready State Diagram

Figure 8-206 “Cable Ready State Diagram” shows the Cable Ready state diagram for a Cable Plug.

Figure 8-206 “Cable Ready State Diagram”



✖ 8.3.3.26.1.1 PE_CBL_Ready State

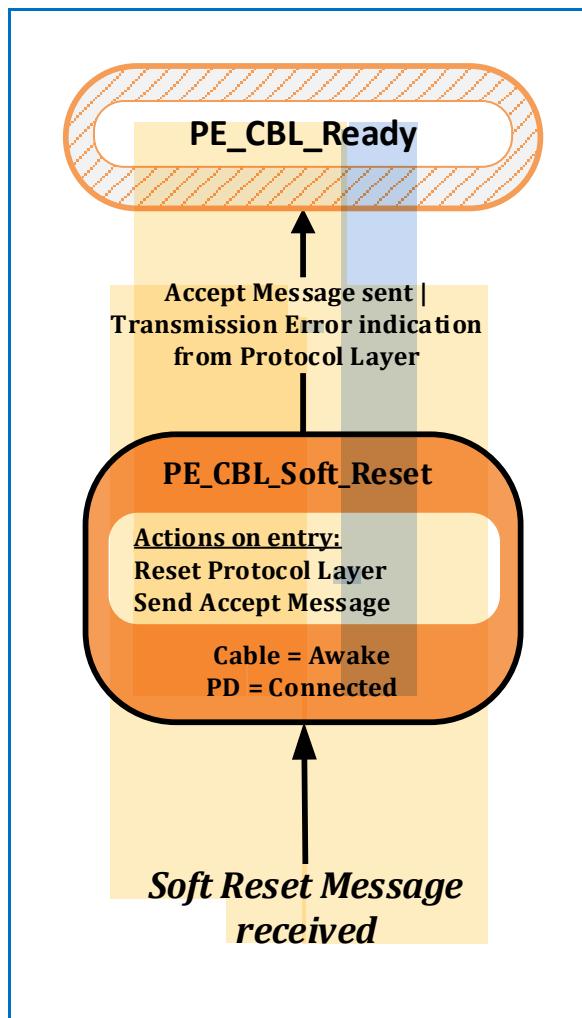
The **PE_CBL_Ready** state shown in the following sections is the normal operational state for a Cable Plug and where it starts after power up or a Hard/Cable Reset.

8.3.3.26.2 Soft/Hard/Cable Reset

8.3.3.26.2.1 Cable Plug Soft Reset State Diagram

Figure 8-207 “Cable Plug Soft Reset State Diagram” shows the Cable Plug state diagram on reception of a **Soft_Reset** Message.

Figure 8-207 “Cable Plug Soft Reset State Diagram”



8.3.3.26.2.1.1 PE_CBL_Soft_Reset State

The **PE_CBL_Soft_Reset** state **Shall** be entered from any state when a Reset Message is received from the Protocol Layer.

On entry to the **PE_CBL_Soft_Reset** state the Policy Engine **Shall** reset the Protocol Layer in the Cable Plug and **Shall** then request the Protocol Layer to send an **Accept** Message.

The Policy Engine **Shall** transition to the **PE_CBL_Ready** state when:

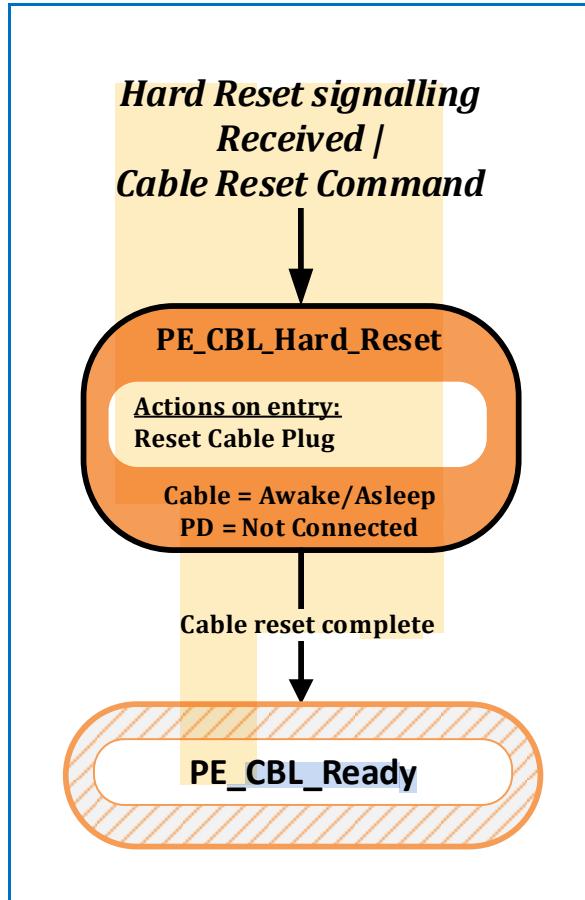
- The **Accept** Message has been sent or
- The Protocol Layer indicates that a transmission error has occurred.

8.3.3.26.2.2

Cable Plug Hard Reset State Diagram

Figure 8-208 “Cable Plug Hard Reset State Diagram” shows the Cable Plug state diagram for a Hard Reset or Cable Reset.

Figure 8-208 “Cable Plug Hard Reset State Diagram”



8.3.3.26.2.2.1

PE_CBL_Hard_Reset State

The **PE_CBL_Hard_Reset** state **Shall** be entered from any state when either **Hard Reset** Signaling or **Cable Reset** Signaling is detected.

On entry to the **PE_CBL_Hard_Reset** state the Policy Engine **Shall** reset the Cable Plug (equivalent to a power cycle).

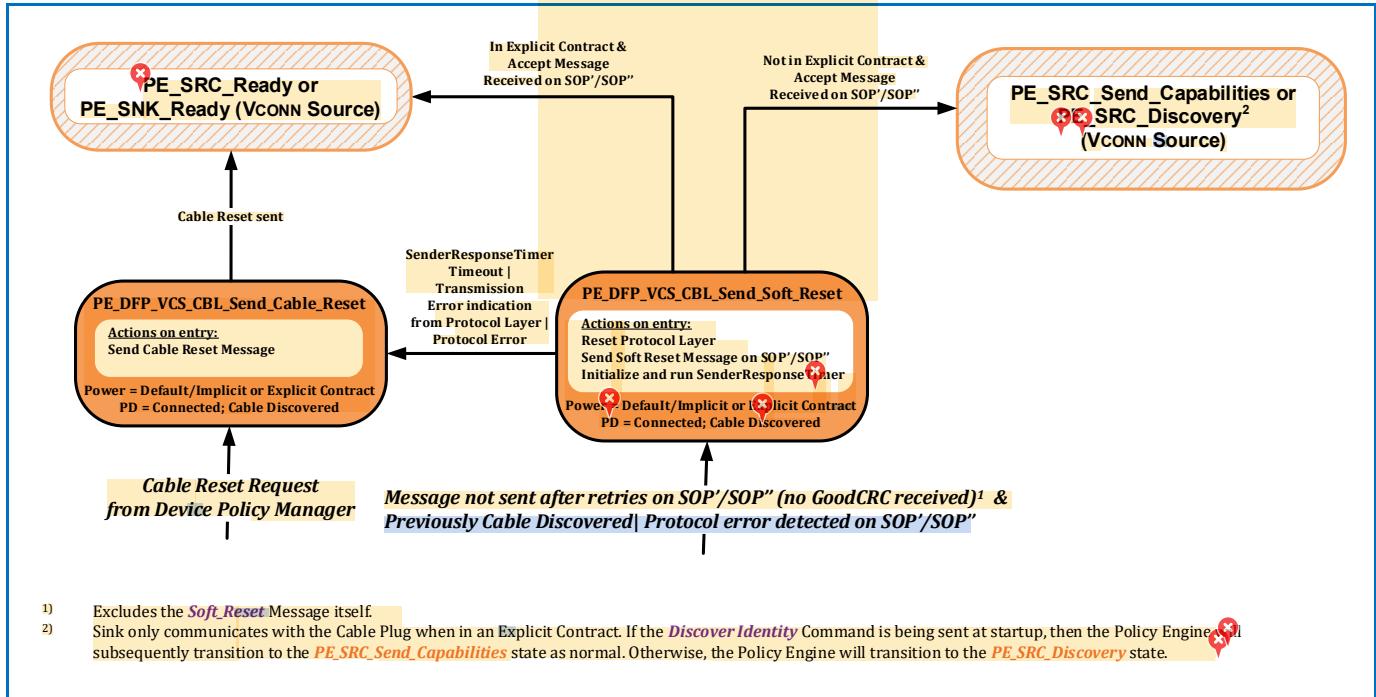
The Policy Engine **Shall** transition to the **PE_CBL_Ready** state when:

- The Cable Plug reset is complete.

8.3.3.26.2.3 DFP/VCONN Source SOP'/SOP" Soft Reset or Cable Reset of a Cable Plug or VPD State Diagram

Figure 8-209 "DFP/Vconn Source Soft Reset or Cable Reset of a Cable Plug or VPD State Diagram" below shows the state diagram for the Policy Engine in a VCONN Source when performing a Soft Reset or Cable Reset of a Cable Plug or VPD on **SOP'/SOP"**. The following sections describe operation in each of the states.

Figure 8-209 "DFP/Vconn Source Soft Reset or Cable Reset of a Cable Plug or VPD State Diagram"



8.3.3.26.2.3.1 PE_DFP_VCS_CBL_Send_Soft_Reset State

The **PE_DFP_VCS_CBL_Send_Soft_Reset** state **Shall** be entered from any state when a Protocol Error is detected on **SOP'/SOP"** by the Protocol Layer (see [Section 6.8.1 "Soft Reset and Protocol Error"](#)) or when a Message has not been sent after retries on **SOP'/SOP"** while communicating with a Cable Plug/VPD and when there was previous communication with the cable plug that did not result in a Transmission Error or whenever the Device Policy Manager directs a Soft Reset on **SOP'/SOP"**.

On entry to the **PE_DFP_VCS_CBL_Send_Soft_Reset** state the DFP Policy Engine **Shall** request the **SOP'/SOP"** Protocol Layer to perform a Soft Reset, then **Shall** send a **Soft_Reset** Message on **SOP'/SOP"** to the Cable Plug/VPD, and initialize and run the **SenderResponseTimer**.

The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state, depending on the DFP VCONN Source's Power Role, when:

- There is no Explicit Contract in place and
- An **Accept** Message has been received on **SOP'/SOP"**.

The Policy Engine **Shall** transition to either the **PE_SRC_Send_Capabilities** state or **PE_SRC_Discovery** state, depending on the DFP's VCONN Source's Power Role, when:

- There is an Explicit Contract in place and
- An **Accept** Message has been received on **SOP'/SOP"**.

The Policy Engine ***Shall*** transition to the ***PE_DFP_VCS_CBL_Send_Cable_Reset*** state when:

- A ***SenderResponseTimer*** timeout occurs
- Or the Protocol Layer indicates that a transmission error has occurred
- Or when a Protocol Error is detected on ***SOP'/SOP''*** by the Protocol Layer.

8.3.3.26.2.3.2 PE_DFP_VCS_CBL_Send_Cable_Reset State

The ***PE_DFP_VCS_CBL_Send_Cable_Reset*** state ***Shall*** be entered from any state when the Device Policy Manager requests a Cable Reset.

On entry to the ***PE_DFP_VCS_CBL_Send_Cable_Reset*** state the DFP Policy Engine ***Shall*** request the Protocol Layer to send ***Cable Reset*** Signaling.

The Policy Engine ***Shall*** transition to either the ***PE_SRC_Ready*** or ***PE_SNK_Ready*** state, depending on the VCONN Source's Power Role, when:

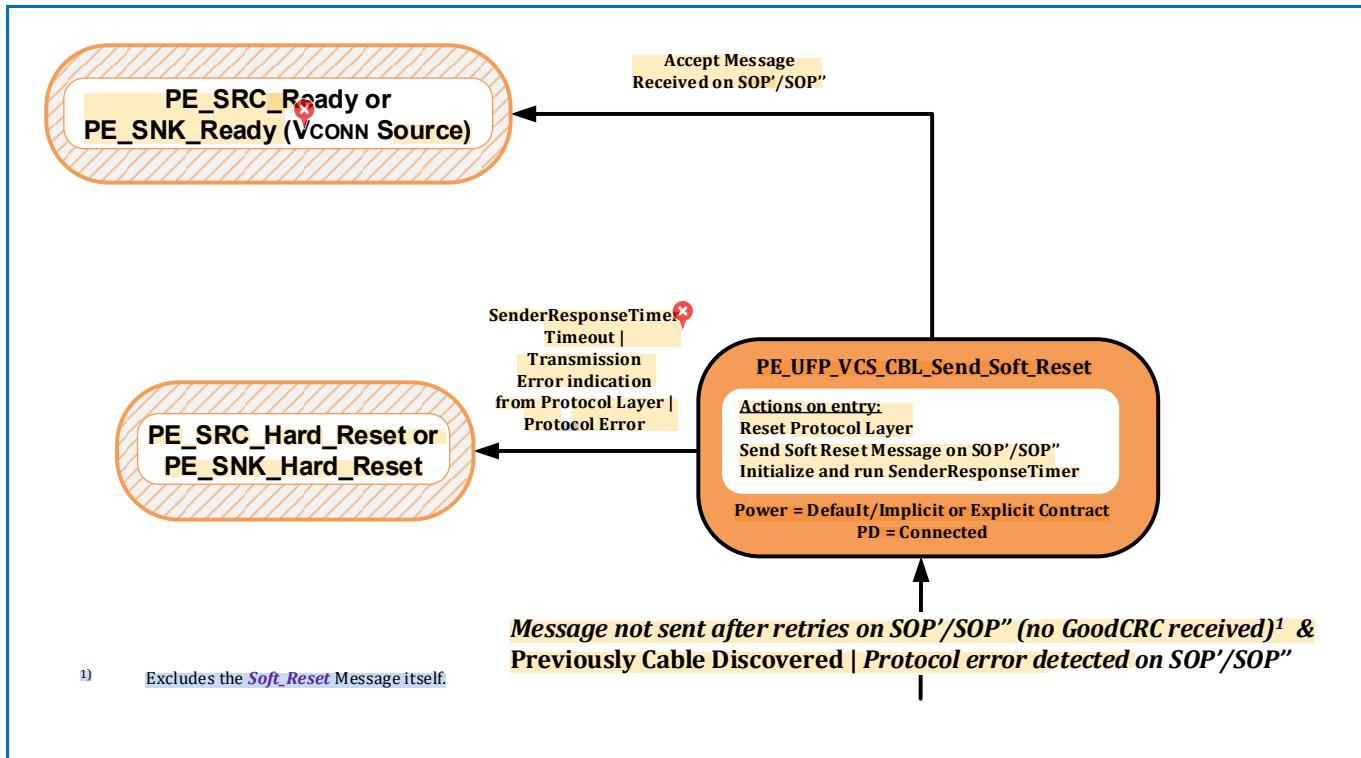
- ***Cable Reset*** Signaling has been sent.

8.3.3.26.2.4

UFP/VCONN Source SOP'/SOP" Soft Reset of a Cable Plug or VPD State Diagram

Figure 8-210 "UFP/Vconn Source Soft Reset of a Cable Plug or VPD State Diagram" below shows the state diagram for the UFP Policy Engine in a VCONN Source when performing a Soft Reset of a Cable Plug or VPD on **SOP'/SOP"**. The following sections describe operation in each of the states.

Figure 8-210 "UFP/VCONN Source Soft Reset of a Cable Plug or VPD State Diagram"



8.3.3.26.2.4.1

PE_UFP_VCS_CBL_Send_Soft_Reset State

The **PE_UFP_VCS_CBL_Send_Soft_Reset** state **Shall** be entered from any state when a Protocol Error is detected on **SOP'/SOP"** by the Protocol Layer (see [Section 6.8.1 "Soft Reset and Protocol Error"](#)) or when a Message has not been sent after retries on **SOP'/SOP"** while communicating with a Cable Plug/VPD and when there was previous communication with the cable plug that did not result in a Transmission Error or whenever the Device Policy Manager directs a Soft Reset on **SOP'/SOP"**.

On entry to the **PE_UFP_VCS_CBL_Send_Soft_Reset** state the Policy Engine **Shall** request the **SOP'/SOP"** Protocol Layer to perform a Soft Reset, then **Shall** send a **Soft_Reset** Message on **SOP'/SOP"** to the Cable Plug, and initialize and run the **SenderResponseTimer**.

The Policy Engine **Shall** transition to either the **PE_SRC_Ready** or **PE_SNK_Ready** state, depending on the UFP VCONN Source's Power Role, when:

- An **Accept** Message has been received on **SOP'/SOP"**.

The Policy Engine **Shall** transition to either the **PE_SRC_Hard_Reset** or **PE_SNK_Hard_Reset** state, depending on the UFP VCONN Source's Power Role, when:

- A **SenderResponseTimer** timeout occurs

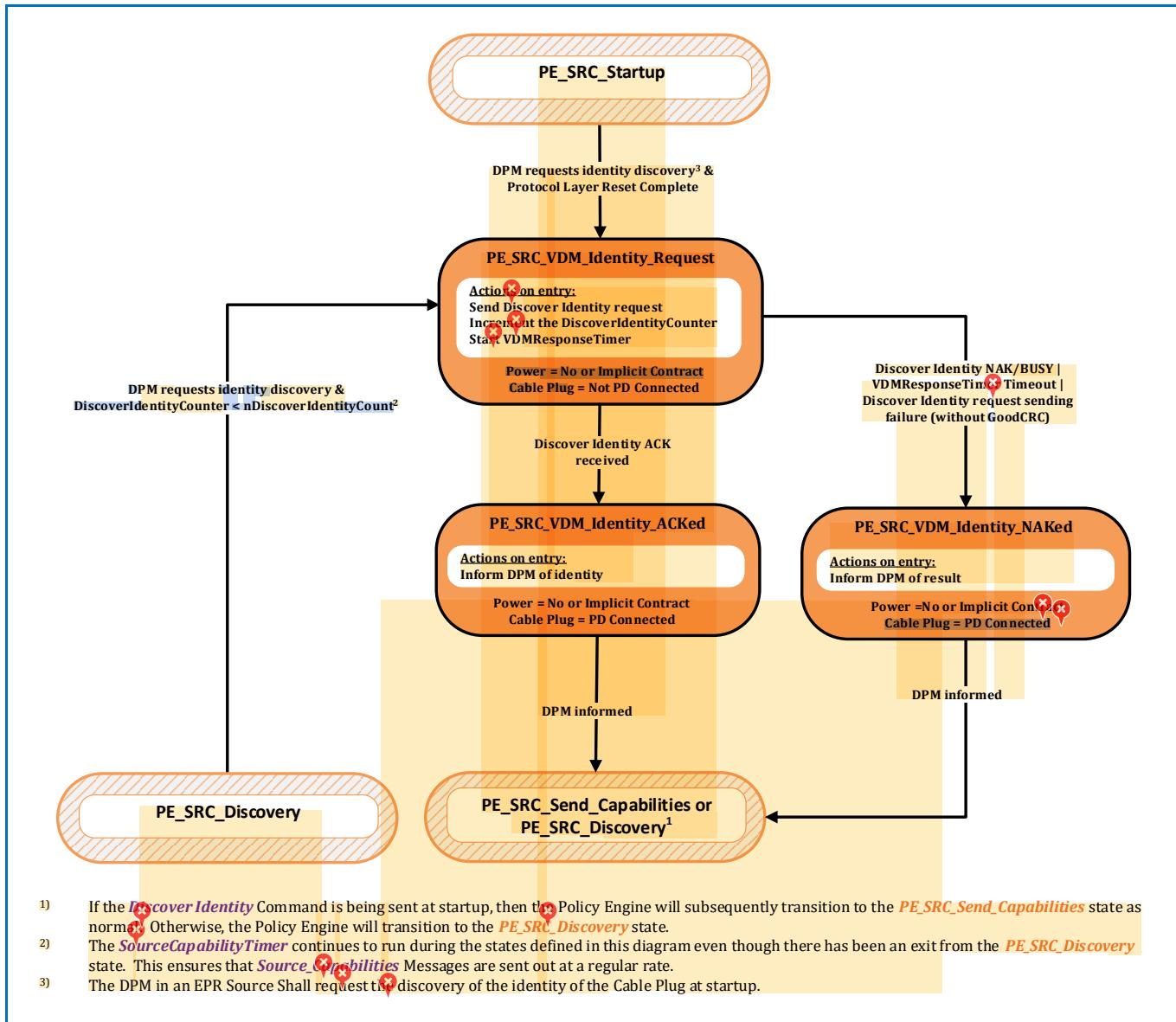
- Or the Protocol Layer indicates that a transmission error has occurred
- Or when a Protocol Error is detected on **SOP'/SOP''** by the Protocol Layer.

8.3.3.26.3

Source Startup Structured VDM Discover Identity of a Cable Plug State Diagram

Figure 8-211 “Source Startup Structured VDM Discover Identity State Diagram” shows the state diagram for Source discovery of identity information from a Cable Plug during the startup sequence.

Figure 8-211 “Source Startup Structured VDM Discover Identity State Diagram”



8.3.3.26.3.1

PE_SRC_VDM_Identity_Request State

The Policy Engine **Shall** transition to the **PE_SRC_VDM_Identity_Request** state from the **PE_SRC_Startup** state when:

- The Device Policy Manager requests the discovery of the identity of the Cable Plug.

The Policy Engine **Shall** transition to the **PE_SRC_VDM_Identity_Request** state from the **PE_SRC_Discovery** state when:

- The Device Policy Manager requests the discovery of the identity of the Cable Plug and 
- The *DiscoverIdentityCounter* < *nDiscoverIdentityCount*.

Even though there has been a transition out of the *PE_SRC_Discovery* state the *SourceCapabilityTimer Shall* continue to run during the states shown in *Figure 8-211 “Source Startup Structured VDM Discover Identity State Diagram”* and *Shall Not* be initialized on re-entry to *PE_SRC_Discovery*.

Note: an EPR Source is required to discover the identity of the Cable Plug prior to entering the First Explicit Contract (see *Section 6.4.10.1 “Process to enter EPR Mode”*)

On entry to the *PE_SRC_VDM_Identity_Request* state the Policy Engine *Shall* send a Structured VDM *Discover Identity* Command request, *Shall* increment the *DiscoverIdentityCounter* and *Shall* start the *VDMResponseTimer*.

The Policy Engine *Shall* transition to the *PE_SRC_VDM_Identity_ACKed* state when:

- A Structured VDM *Discover Identity* ACK Command response is received.

The Policy Engine *Shall* transition to the *PE_SRC_VDM_Identity_NAKed* state when:

- A Structured VDM *Discover Identity* NAK or BUSY Command response is received or
- The *VDMResponseTimer* times out or
- The Structured VDM *Discover Identity* Command request Message sending fails (no *GoodCRC* Message received after retries).

8.3.3.26.3.2 PE_SRC_VDM_Identity_ACKed State

On entry to the *PE_SRC_VDM_Identity_ACKed* state the Policy Engine *Shall* inform the Device Policy Manager of the Identity information.

The Policy Engine *Shall* transition back to either the *PE_SRC_Send_Capabilities* or *PE_SRC_Discovery* state when:

- The Device Policy Manager has been informed.

8.3.3.26.3.3 PE_SRC_VDM_Identity_NAKed State

On entry to the *PE_SRC_VDM_Identity_NAKed* state the Policy Engine *Shall* inform the Device Policy Manager of the result (NAK, BUSY or timeout).

The Policy Engine *Shall* transition back to either the *PE_SRC_Send_Capabilities* or *PE_SRC_Discovery* state when:

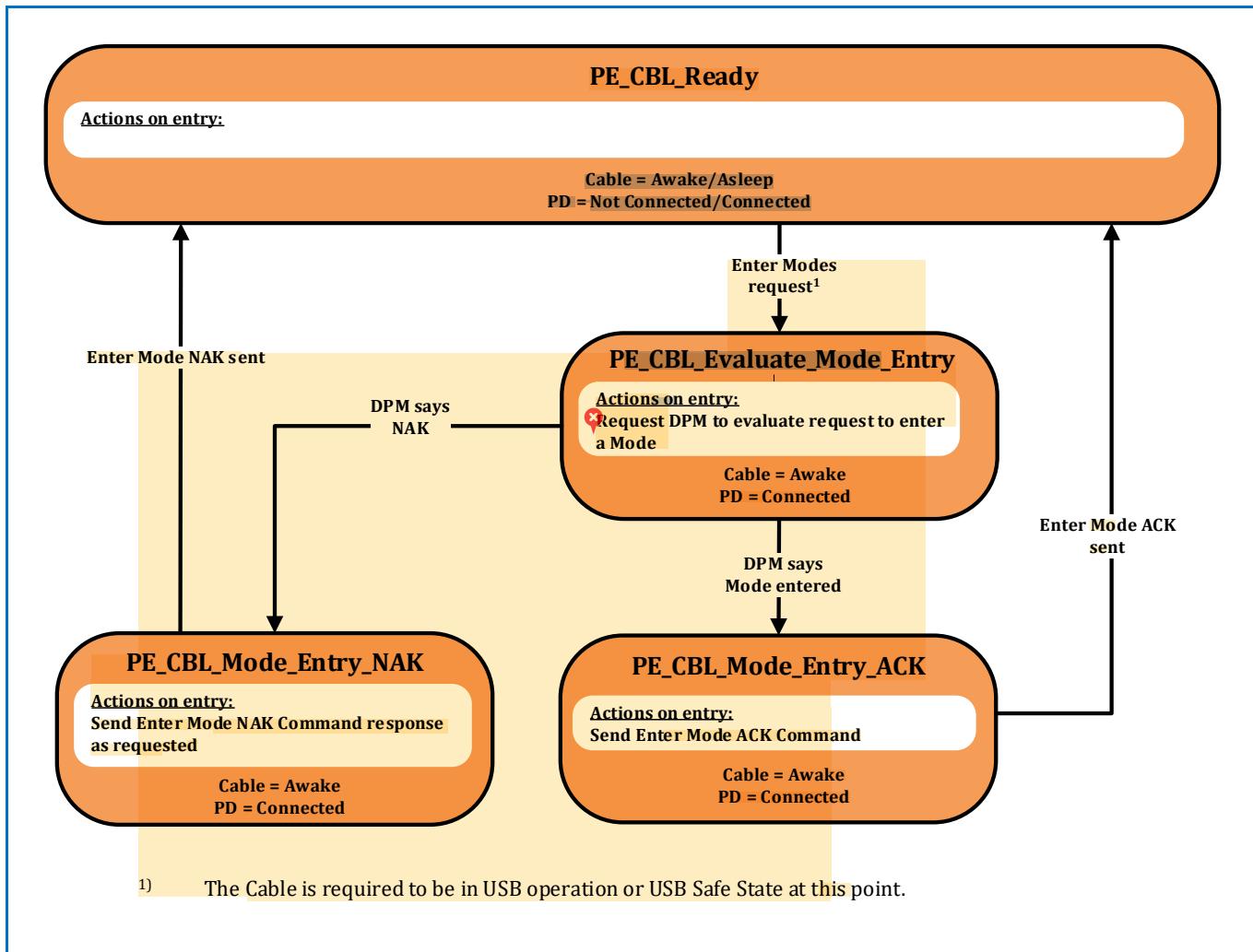
- The Device Policy Manager has been informed.

8.3.3.26.4 Cable Plug Mode Entry/Exit

8.3.3.26.4.1 Cable Plug Structured VDM Enter Mode State Diagram

Figure 8-212 “Cable Plug Structured VDM Enter Mode State Diagram” shows the state diagram for a Cable Plug in response to an **Enter Mode** Command.

Figure 8-212 “Cable Plug Structured VDM Enter Mode State Diagram”



8.3.3.26.4.1.1 **PE_CBL_Evaluate_Mode_Entry State**

 The Policy Engine transitions to the **PE_CBL_Evaluate_Mode_Entry** state from the **PE_CBL_Ready** state when:

- A Structured VDM **Enter Mode** Command request is received from the DFP.

On Entry to the **PE_CBL_Evaluate_Mode_Entry** state the Policy Engine **Shall** request the Device Policy Manager to evaluate the **Enter Mode** Command request and enter the Mode indicated in the Command request if the request is acceptable.

The Policy Engine **Shall** transition to the **PE_CBL_Mode_Entry_ACK** state when:

- The Device Policy Manager indicates that the Mode has been entered.

The Policy Engine ***Shall*** transition to the ***PE_CBL_Mode_Entry_NAK*** state when:

- The Device Policy Manager indicates that the response to the Mode request is NAK.

8.3.3.26.4.1.2 PE_CBL_Mode_Entry_ACK State

On entry to the ***PE_CBL_Mode_Entry_ACK*** state the Policy Engine ***Shall*** send a Structured VDM ***Enter Mode ACK*** Command response.

The Policy Engine ***Shall*** transition to the ***PE_CBL_Ready*** state when:

- The Structured VDM ***Enter Mode*** ACK Command response has been sent.

8.3.3.26.4.1.3 PE_CBL_Mode_Entry_NAK State

On entry to the ***PE_CBL_Mode_Entry_NAK*** state the Policy Engine ***Shall*** send a Structured VDM ***Enter Mode NAK*** Command response as indicated by the Device Policy Manager.

The Policy Engine ***Shall*** transition to the ***PE_CBL_Ready*** state when:

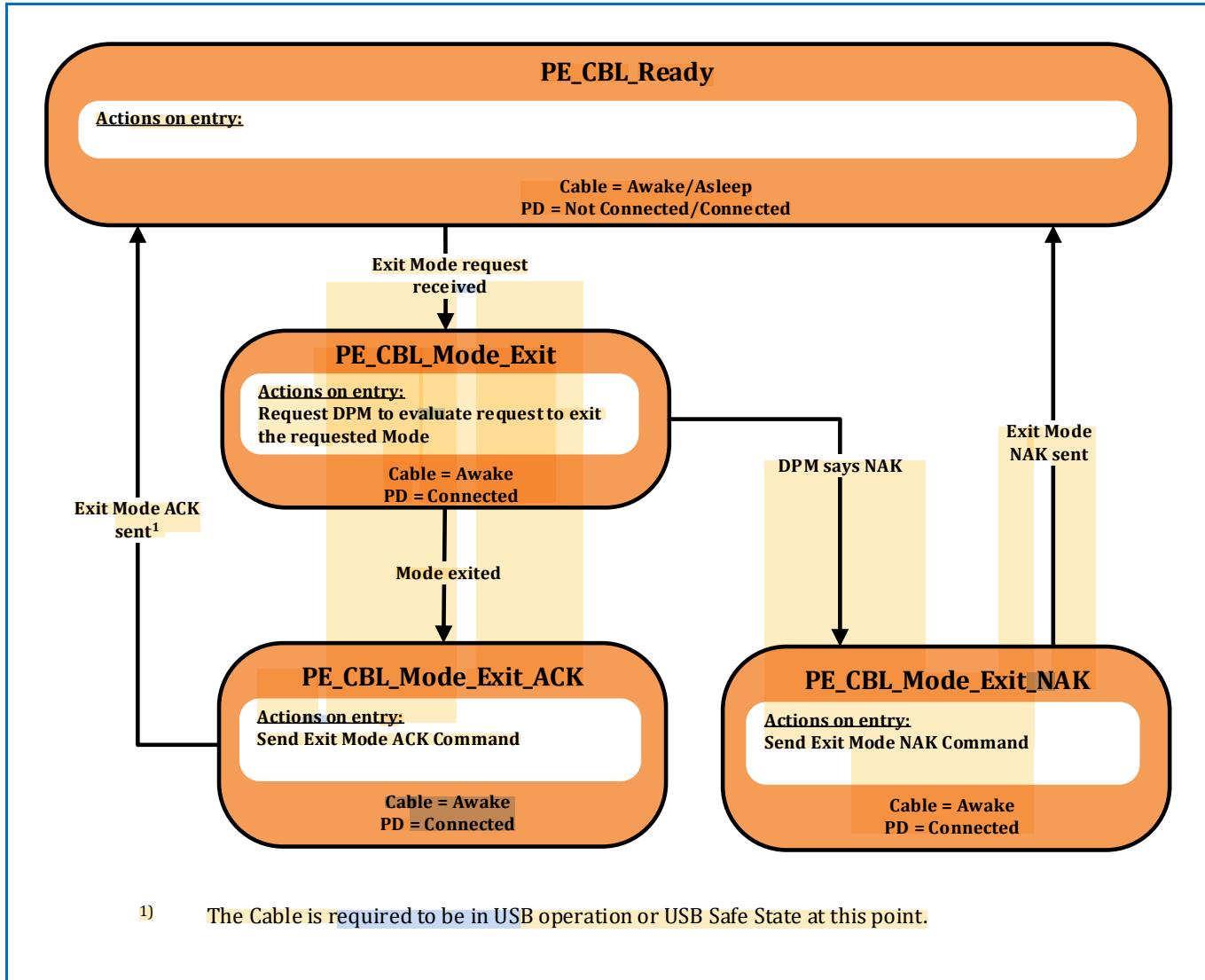
- The Structured VDM ***Enter Mode*** NAK Command response has been sent.

8.3.3.26.4.2

Cable Plug Structured VDM Exit Mode State Diagram

Figure 8-213 “Cable Plug Structured VDM Exit Mode State Diagram” shows the state diagram for a Cable Plug in response to an **Exit Mode** Command.

Figure 8-213 “Cable Plug Structured VDM Exit Mode State Diagram”



8.3.3.26.4.2.1

PE_CBL_Mode_Exit State

The Policy Engine transitions to the **PE_CBL_Mode_Exit** state from the **PE_CBL_Ready** state when:

- A Structured VDM **Exit Mode** Command request is received from the DFP.

On entry to the **PE_CBL_Mode_Exit** state the Policy Engine **Shall** request the Device Policy Manager to exit the Mode indicated in the Command.

The Policy Engine **Shall** transition to the **PE_CBL_Mode_Exit_ACK** state when:

- The Device Policy Manager indicates that the Mode has been exited.

The Policy Engine **Shall** transition to the **PE_CBL_Mode_Exit_NAK** state when:

- The Device Policy Manager indicates that the Command response to the **Exit Mode** Command request is NAK.

8.3.3.26.4.2.2 PE_CBL_Mode_Exit_ACK State

On entry to the **PE_CBL_Mode_Exit_ACK** state the Policy Engine **Shall** send a Structured VDM **Exit Mode** ACK Command response.

The Policy Engine **Shall** transition to the **PE_CBL_Ready** state when:

- The Structured VDM **Exit Mode** ACK Command response has been sent.

8.3.3.26.4.2.3 PE_CBL_Mode_Exit_NAK State

On entry to the **PE_CBL_Mode_Exit_NAK** state the Policy Engine **Shall** send a Structured VDM **Exit Mode** NAK Command response as indicated by the Device Policy Manager.

The Policy Engine **Shall** transition to the **PE_CBL_Ready** state when:

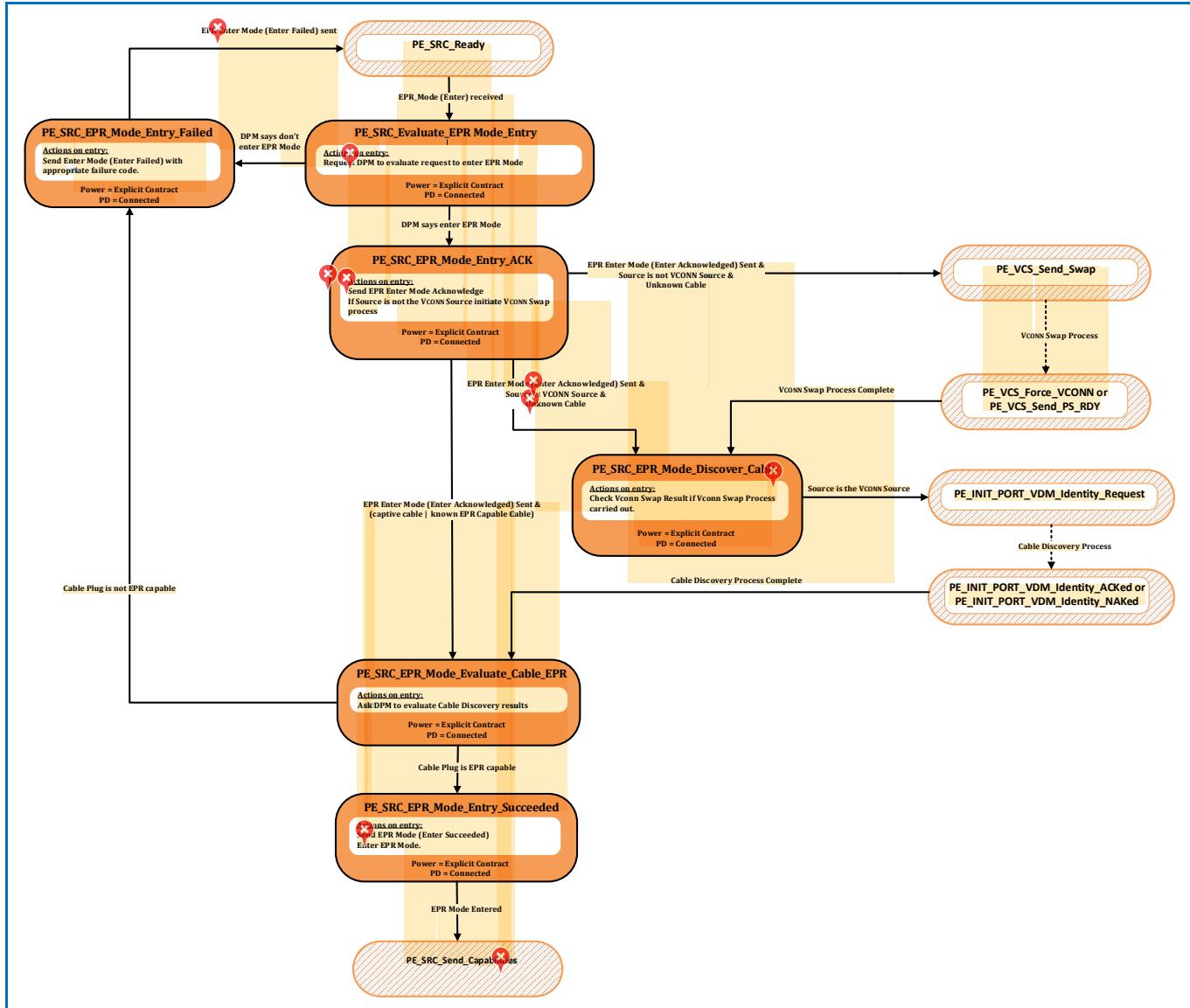
- The Structured VDM **Exit Mode** NAK Command response has been sent.

8.3.3.27 EPR Mode State Diagrams

8.3.3.27.1 Source EPR Mode Entry State Diagram

Figure 8-214 “Source EPR Mode Entry State Diagram” shows the state diagram for an EPR Source in response to an **EPR_Mode** Message.

Figure 8-214 “Source EPR Mode Entry State Diagram”



8.3.3.27.1.1 PE_SRC_Evaluate_EPR_Mode_Entry State

• The Policy Engine transitions to the **PE_SRC_Evaluate_EPR_Mode_Entry** state from the **PE_SRC_Ready** state when:

- An **EPR_Mode** (Enter) Message is received from the Sink.

On Entry to the **PE_SRC_Evaluate_EPR_Mode_Entry** state the Policy Engine **Shall** request the Device Policy Manager to evaluate the **EPR_Mode** (Enter) Message.

 The Policy Engine **Shall** transition to the **PE_SRC_EPR_Mode_Entry_Ack** state when:

- The Device Policy Manager indicates that EPR Mode can be entered.

The Policy Engine **Shall** transition to the **PE_SRC_EPR_Mode_Entry_Failed** state when:

- The Device Policy Manager indicates that the EPR Mode is not to be entered.

8.3.3.27.1.2 PE_SRC_EPR_Mode_Entry_Ack State

On entry to the **PE_SRC_EPR_Mode_Entry_Ack** state the Policy Engine **Shall** send a **EPR_Mode** (Enter Acknowledged) Message. 

The Policy Engine **Shall** transition to the **PE_SRC_EPR_Mode_Evaluate_Cable_EPR** state when:

- The **EPR_Mode** (Enter Acknowledged) Message has been sent and
- The Source is not the VCONN Source and
- The cable is a captive cable or a known EPR Capable Cable.

The Policy Engine **Shall** transition to the **PE_VCS_Send_Swap** state when:

- The **EPR_Mode** (Enter Acknowledged) Message has been sent and
- The Source is not the VCONN Source and
- The cable is unknown.

The Policy Engine **Shall** transition to the **PE_SRC_EPR_Mode_Discover_Cable** state when:

- The **EPR_Mode** (Enter Acknowledged) Message has been sent and
- The Source is the VCONN Source and
- The cable is unknown.

8.3.3.27.1.3 PE_SRC_EPR_Mode_Discover_Cable State

The Policy Engine transitions to the **PE_SRC_EPR_Mode_Discover_Cable** state from the **PE_VCS_Force_VCONN** state or **PE_VCS_Send_Ps_Rdy** state when:

- A Source initiated VCONN Swap process has completed.

The Policy Engine **Shall** transition to the **PE_INIT_PORT_VDM_Identity_Request** state in order to perform Cable Plug discovery when:

- The Source is the VCONN Source.

The Policy Engine **Shall** transition to the **PE_SRC_EPR_Mode_Entry_Failed** state when:

- The VCONN Swap process failed (the Source is not the VCONN Source).

8.3.3.27.1.4 PE_SRC_EPR_Mode_Evaluate_Cable_EPR State

In the state the Policy Engine requests the DPM to evaluate the Cable Discovery results.

The Policy Engine **Shall** transition to the **PE_SRC_EPR_Mode_Entry_Succeeded** state when:

- The Cable Plug is capable of EPR Mode.

The Policy Engine **Shall** transition to the **PE_SRC_EPR_Mode_Entry_Failed** state when:

- The Cable Plug is not capable of EPR Mode.

✖ 8.3.3.27.1.5 PE_SRC_EPR_Mode_Entry_Succeeded State

On entry to the **PE_SRC_EPR_Mode_Entry_Succeeded** state the Policy Engine **Shall** send a **EPR_Mode** (Enter Succeeded) Message and enter EPR Mode.

The Policy Engine **Shall** transition to the **PE_SRC_Send_Capabilities** state when:

- EPR Mode has been entered.

8.3.3.27.1.6 PE_SRC_EPR_Mode_Entry_Failed State

On entry to the **PE_SRC_EPR_Mode_Entry_Failed** state the Policy Engine **Shall** send a **EPR_Mode** (Enter Failed) Message.

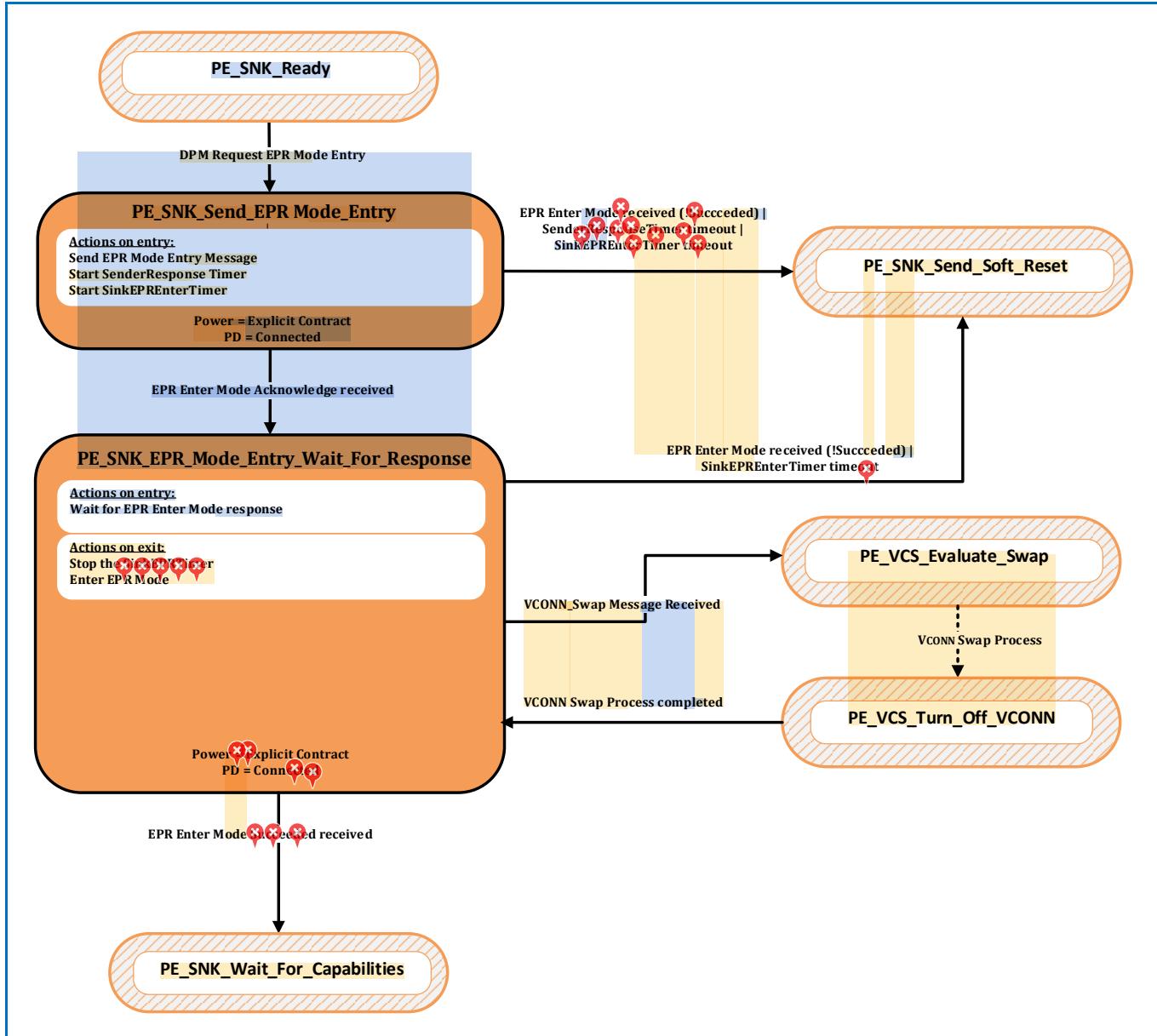
The Policy Engine **Shall** transition to the **PE_SRC_Ready** state when:

- The **EPR_Mode** (Enter Failed) Message has been sent.

8.3.3.27.2 Sink EPR Mode Entry State Diagram

Figure 8-215 “Sink EPR Mode Entry State Diagram” shows the state diagram for an EPR Sink initiating the EPR Mode Entry process.

Figure 8-215 “Sink EPR Mode Entry State Diagram”



8.3.3.27.2.1 PE_SNK_Send_EPR_Mode_Entry State

The Policy Engine transitions to the **PE_SNK_Send_EPR_Mode_Entry** state from the **PE_SNK_Ready** state when:

- The DPM requests entry into EPR Mode.

On Entry to the **PE_SNK_Send_EPR_Mode_Entry** state the Policy Engine **Shall** send an **EPR_Mode** (Enter) Message and starts the **SenderResponseTimer** and the **SinkEPREnterTimer**. Note that the **SinkEPREnterTimer** **Shall** continue to run in every state until it is stopped or times out.

The Policy Engine ***Shall*** transition to the ***PE_SNK_EPR_Mode_Wait_For_Response*** state when:

- An ***EPR_Mode*** (Enter Acknowledge) Message is received.

✖ The Policy Engine ***Shall*** transition to the ***PE_SNK_Send_Soft_Reset*** state when:

- An ***EPR_Mode*** Message is received which is not Enter Succeeded or
- The ***SenderResponseTimer*** times out or
- The ***SinkEPREnEnterTimer*** times out.

8.3.3.27.2.2 PE_SNK_EPR_Mode_Wait_For_Response State

In the State the Policy Engine waits for a confirmation that the EPR Mode entry request has succeeded.

On exit from the ***PE_SNK_EPR_Mode_Wait_For_Response*** state the Policy Engine ***Shall*** stop the ***SinkEPREnEnterTimer*** and enter EPR Mode.

The Policy Engine ***Shall*** transition to the ***PE_SNK_Send_Soft_Reset*** state when:

- An ***EPR_Mode*** Message is received which is not Enter Succeeded or
- The ***SinkEPREnEnterTimer*** times out.

The Policy Engine ***Shall*** transition to the ***PE_VCS_Evaluate_Swap*** State when:

- A ***VCONN_Swap*** Message is received.

The Policy Engine ***Shall*** transition back from the ***PE_VCS_Turn_Off_VCONN*** State to the ***PE_SNK_EPR_Mode_Wait_For_Response*** State when:

- The VCONN Swap process has completed.

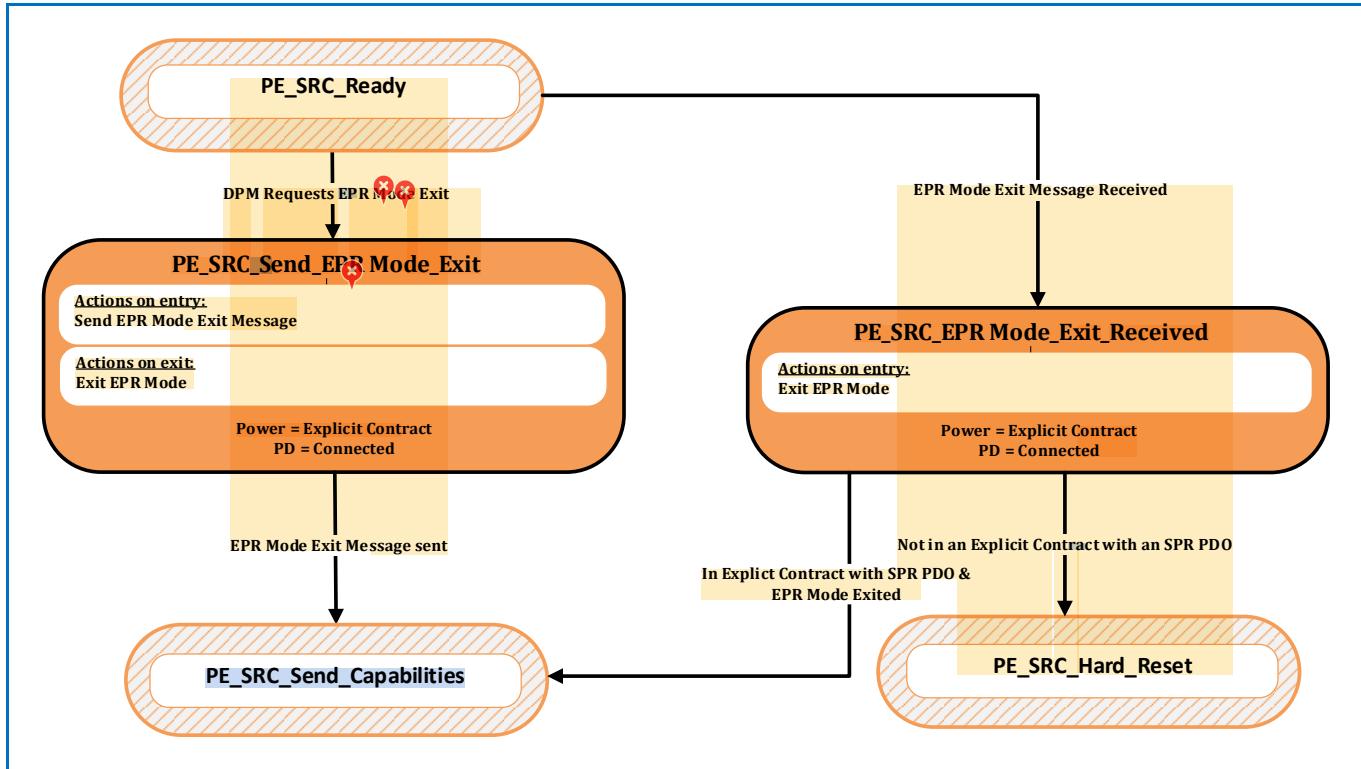
The Policy Engine ***Shall*** transition to the ***PE_SNK_Wait_for_Capabilities*** state when:

- An ***EPR_Mode*** (Enter Succeeded) Message has been received.

8.3.3.27.3 | Source EPR Mode Exit State Diagram

Figure 8-216 “Source EPR Mode Exit State Diagram” shows the state diagram for an EPR Source initiating the EPR Mode exit process.

Figure 8-216 “Source EPR Mode Exit State Diagram”



8.3.3.27.3.1 PE_SRC_Send_EPR_Mode_Exit State

The Policy Engine transitions to the **PE_SRC_Send_EPR_Mode_Exit** state from the **PE_SRC_Ready** state when:

- The DPM requests exit from EPR Mode.

On Entry to the **PE_SRC_Send_EPR_Mode_Exit** state the Policy Engine **Shall** send an **EPR_Mode** (Exit) Message.

On Exit from the **PE_SRC_Send_EPR_Mode_Exit** state the Policy Engine **Shall** exit EPR Mode.

The Policy Engine **Shall** transition to the **PE_SRC_Send_Capabilities** state when:

- The **EPR_Mode** (Exit) Message has been sent.

8.3.3.27.3.2 PE_SRC_EPR_Mode_Exit_Received State

The Policy Engine transitions to the **PE_SRC_EPR_Mode_Exit_Received** state from the **PE_SRC_Ready** state when:

- An **EPR_Mode** (Exit) Message is received.

On Entry to the **PE_SRC_EPR_Mode_Exit_Received** state the Policy Engine **Shall** exit EPR Mode.

The Policy Engine **Shall** transition to the **PE_SRC_Send_Capabilities** state when:

- In an Explicit Contract with an SPR PDO and

- EPR Mode has been exited.

The Policy Engine ***Shall*** transition to the ***PE_SRC_Hard_Reset*** state when:

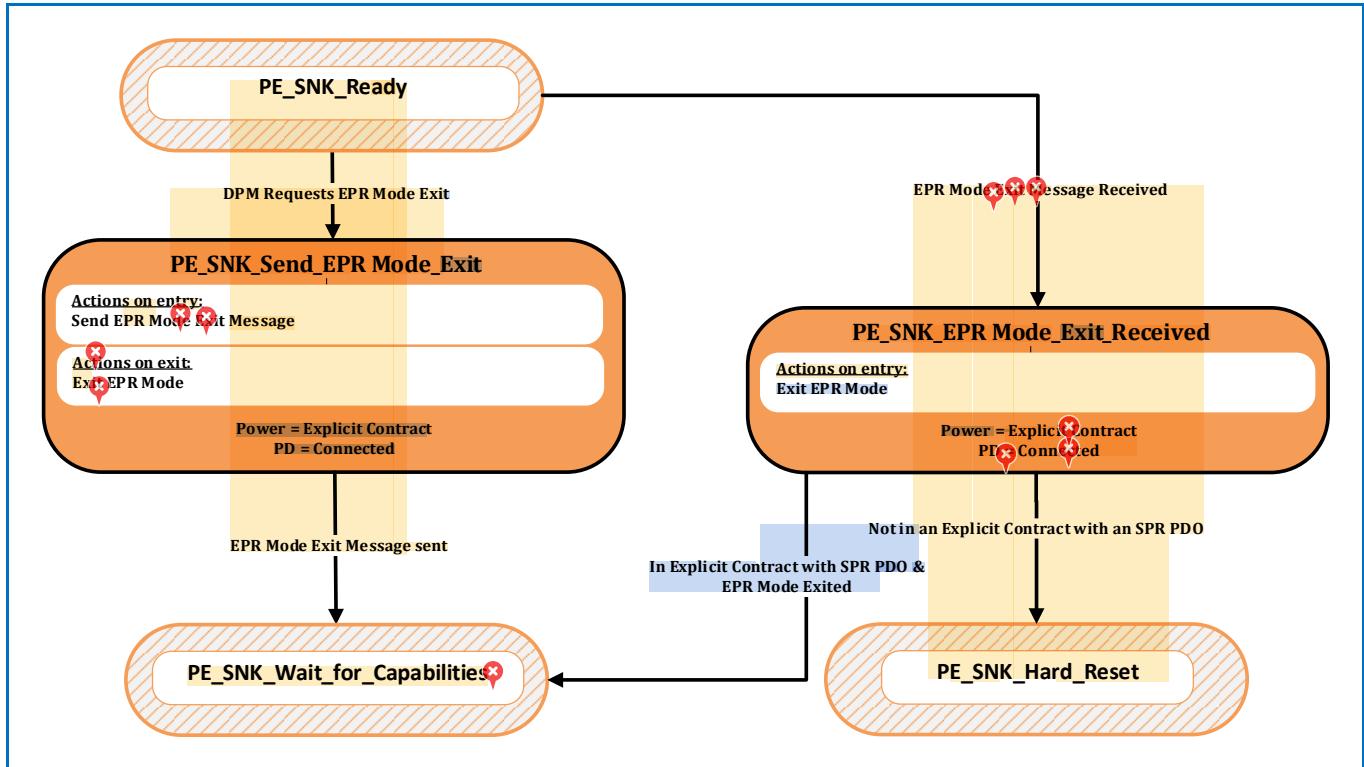
- Not in an Explicit Contract with an SPR PDO.

8.3.3.27.4

Sink EPR Mode Exit State Diagram

Figure 8-217 “Sink EPR Mode Exit State Diagram” shows the state diagram for an EPR Sink initiating the EPR Mode exit process.

Figure 8-217 “Sink EPR Mode Exit State Diagram”



8.3.3.27.4.1

PE_SNK_Send_EPR_Mode_Exit State

The Policy Engine transitions to the **PE_SNK_Send_EPR_Mode_Exit** state from the **PE_SNK_Ready** state when:

- The DPM requests exit from EPR Mode.

On Entry to the **PE_SNK_Send_EPR_Mode_Exit** state the Policy Engine **Shall** send an **EPR_Mode** (Exit) Message.

On Exit from the **PE_SNK_Send_EPR_Mode_Exit** state the Policy Engine **Shall** exit EPR Mode.

The Policy Engine **Shall** transition to the **PE_SNK_Wait_for_Capabilities** state when:

- The **EPR_Mode** (Exit) Message has been sent.

8.3.3.27.4.2

PE_SNK_EPR_Mode_Exit_Received State

The Policy Engine transitions to the **PE_SNK_EPR_Mode_Exit_Received** state from the **PE_SNK_Ready** state when:

- An **EPR_Mode** (Exit) Message is received.

On Entry to the **PE_SNK_EPR_Mode_Exit_Received** state the Policy Engine **Shall** exit EPR Mode.

The Policy Engine **Shall** transition to the **PE_SNK_Wait_for_Capabilities** state when:

- In an Explicit Contract with an SPR PDO and

- EPR Mode has been exited.

The Policy Engine ***Shall*** transition to the ***PE_SNK_Hard_Reset*** state when:

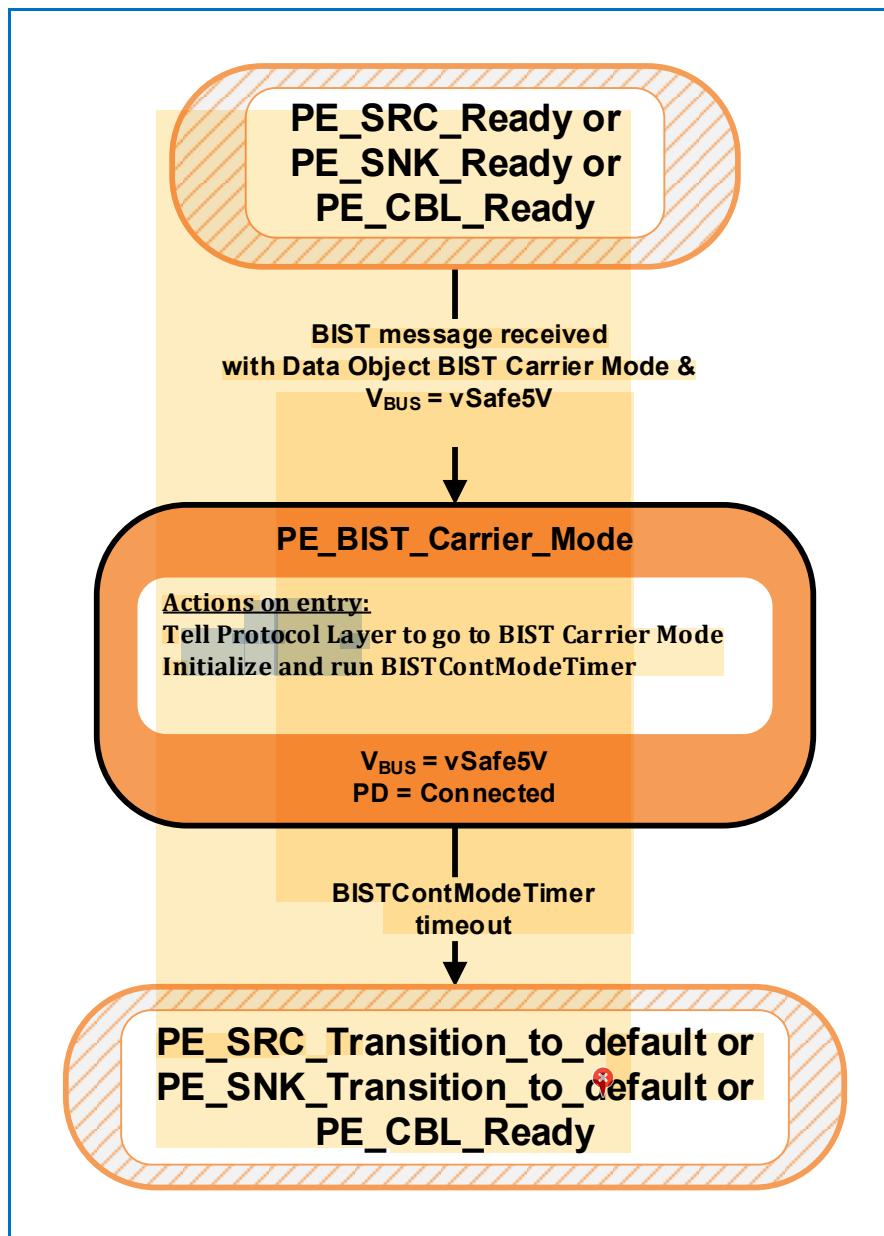
- Not in an Explicit Contract with an SPR PDO.

8.3.3.28 BIST State diagrams

8.3.3.28.1 BIST Carrier Mode State Diagram

Figure 8-218 “BIST Carrier Mode State Diagram” shows the state diagram required by a UUT, which can be either a Source, Sink or Cable Plug, when operating in BIST Carrier Mode. Transitions **Shall** be from either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** states.

Figure 8-218 “BIST Carrier Mode State Diagram”



8.3.3.28.1.1 PE_BIST_Carrier_Mode State

The Source, Sink or Cable Plug **Shall** enter the **PE_BIST_Carrier_Mode** state from either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state when:

- A **BIST** Message is received with a **BIST Carrier Mode** BIST Data Object and
- V_{BUS} is at **vSafe5V**.

On entry to the **PE_BIST_Carrier_Mode** state the Policy Engine **Shall** tell the Protocol Layer to go to BIST Carrier Mode (see [Section 6.4.3.1 "BIST Carrier Mode"](#)) and **Shall** initialize and run the **BISTContModeTimer**.

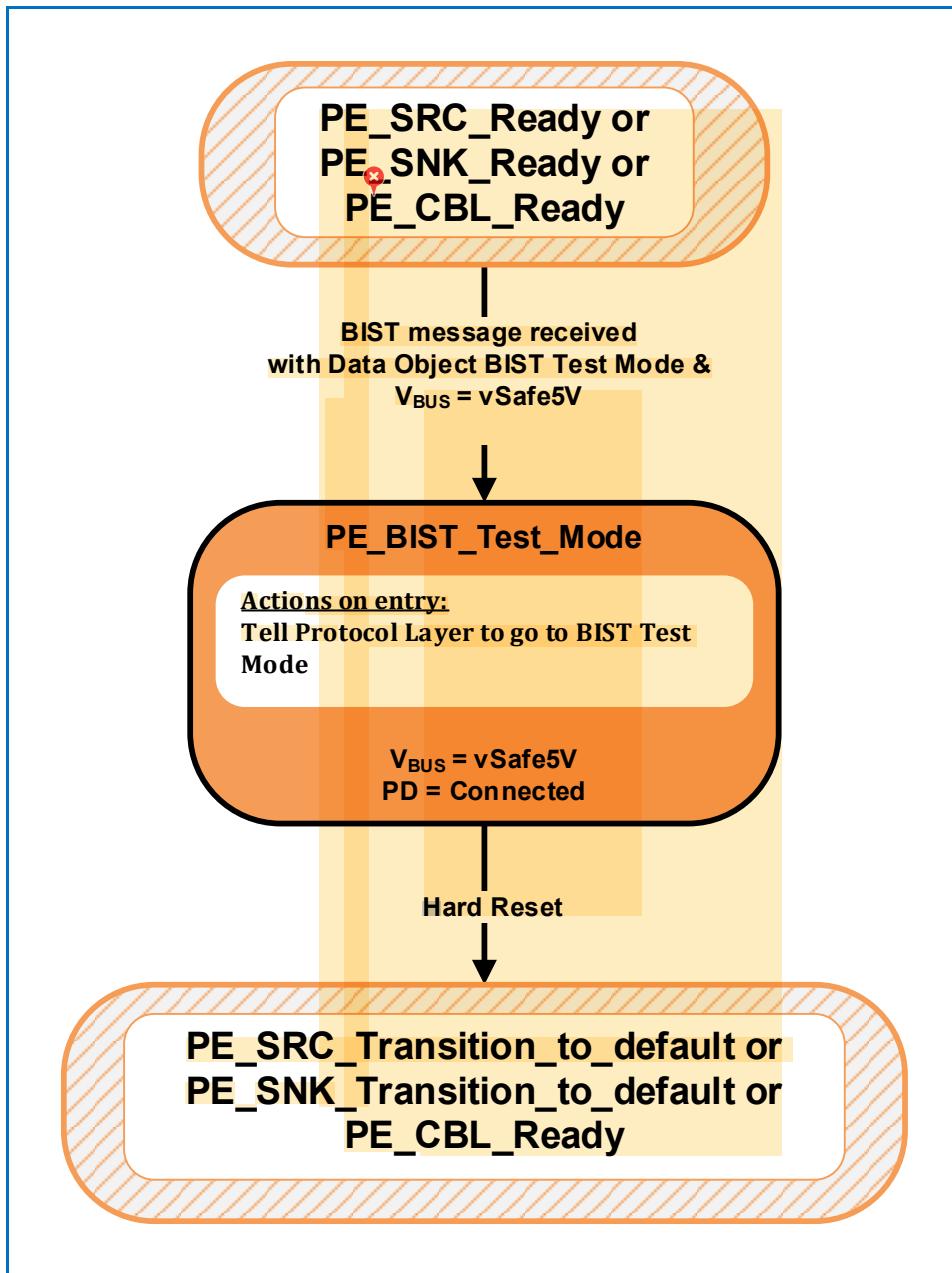
The Policy Engine **Shall** transition to either the **PE_SRC_Transition_to_default** state, **PE_SNK_Transition_to_default** state or **PE_CBL_Ready** state (as appropriate) when:

- The **BISTContModeTimer** times out.

8.3.3.28.2 BIST Test Mode State Diagram

Figure 8-219 “BIST Test Mode State Diagram” shows the state diagram required by a UUT, which can be either a Source, Sink or Cable Plug, when operating in BIST Test Data Mode. Transitions *Shall* be from either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** states.

Figure 8-219 “BIST Test Mode State Diagram”



8.3.3.28.2.1 PE_BIST_Test_Mode State

The Source, Sink or Cable Plug *Shall* enter the **PE_BIST_Test_Mode** state from either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state when:

- A **BIST** Message is received with a **BIST Test Data** BIST Data Object and

- V_{BUS} is at **vSafe5V**.

On entry to the **PE_BIST_Test_Mode** state the Policy Engine **Shall** tell the Protocol Layer to go to BIST Test Mode and **Shall** into BIST Test Data Mode where it sends no further Messages except for **GoodCRC** Messages in response to received Messages (see [Section 6.4.3.2 "BIST Test Data"](#)).

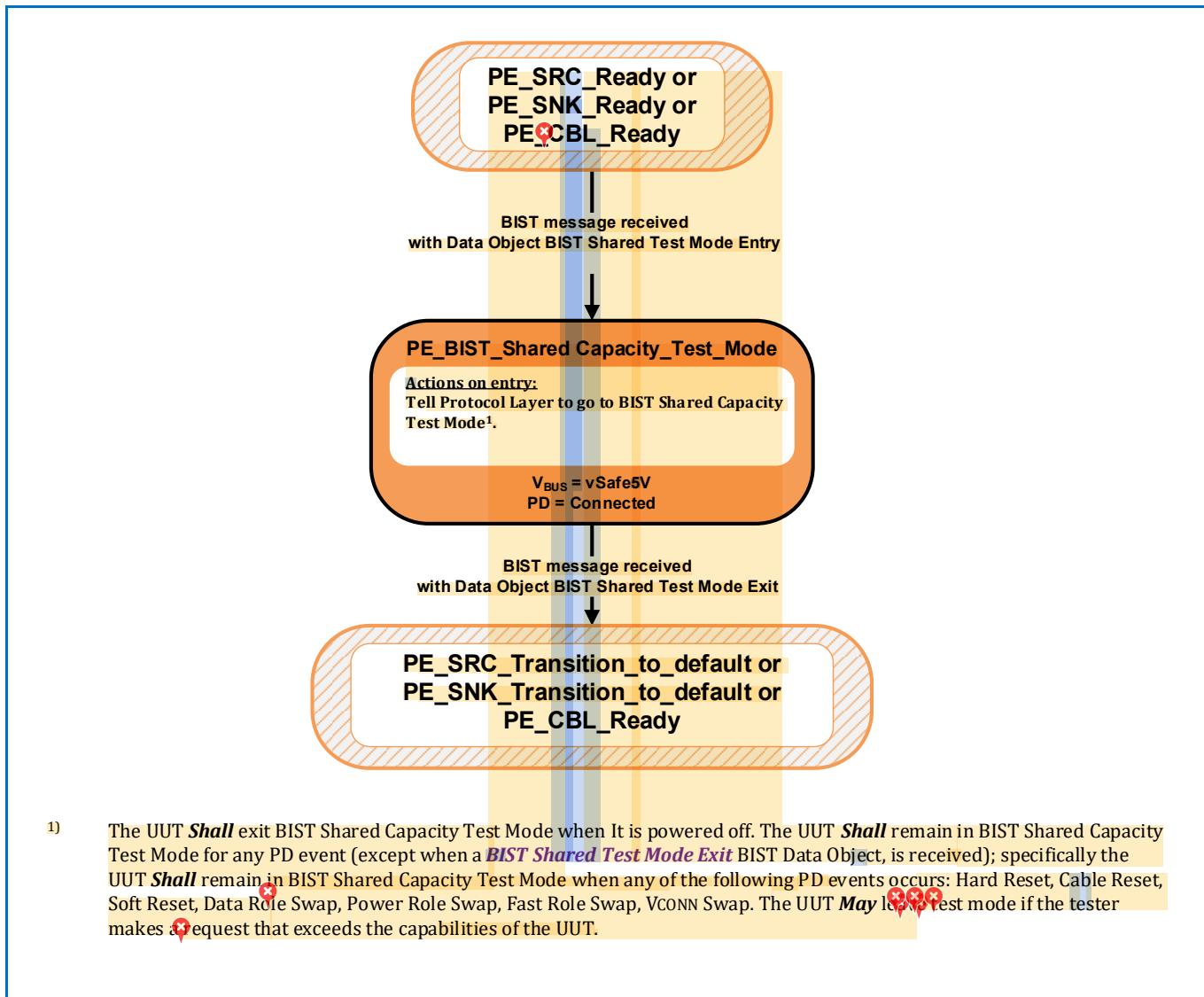
The Policy Engine **Shall** transition to either the **PE_SRC_Transition_to_default** state, **PE_SNK_Transition_to_default** state or **PE_CBL_Ready** state (as appropriate) when:

- A Hard Reset occurs.

8.3.3.28.3 BIST Shared Capacity Test Mode State Diagram

Figure 8-220 “BIST Shared Capacity Test Mode State Diagram” shows the state diagram required by a UUT, which can be either a Source, Sink or Cable Plug, when operating in BIST Shared Capacity Test Mode. Transitions **Shall** be from either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** states.

Figure 8-220 “BIST Shared Capacity Test Mode State Diagram”



8.3.3.28.3.1 PE_BIST_Shared_Capacity_Test_Mode State

The Source, Sink or Cable Plug **Shall** enter the **PE_BIST_Shared_Capacity_Test_Mode** state from either the **PE_SRC_Ready**, **PE_SNK_Ready** or **PE_CBL_Ready** state when:

- A **BIST** Message is received with a **BIST Shared Test Mode Entry** BIST Data Object and
- V_{BUS} is at **vSafe5V**.

On entry to the **PE_BIST_Shared_Capacity_Test_Mode** state the Policy Engine **Shall** tell the Protocol Layer to go to BIST Shared Capacity Test Mode (see [Section 6.4.3.3 “BIST Shared Capacity Test Mode”](#)).

The Policy Engine ***Shall*** transition to either the ***PE_SRC_Transition_to_default*** state, ***PE_SNK_Transition_to_default*** state or ***PE_CBL_Ready*** state (as appropriate) when:

- A ***BIST*** Message is received with a ***BIST Shared Test Mode Exit*** BIST Data Object.

8.3.3.29 USB Type-C® Referenced States

This section contains states cross-referenced from the [\[USB Type-C 2.3\]](#) specification.

8.3.3.29.1 ErrorRecovery state

The **ErrorRecovery** state is used to electronically disconnect Port Partners using the USB Type-C® connector. The **ErrorRecovery** state **Shall** be entered when there are errors on USB Type-C® Ports which cannot be recovered by Hard Reset. The **ErrorRecovery** state **Shall** map to USB Type-C® ErrorRecovery state operation as defined in the [\[USB Type-C 2.3\]](#) specification. Bus powered Sinks **Shall Not** be required to meet this requirement as removal of their power will serve the same purpose.

On entry to the **ErrorRecovery** state the Contract and PD Connection **Shall** be ended.

On exit from the **ErrorRecovery** state a new Explicit Contract **Should** be established once the Port Partners have re-connected over the CC wire.

8.3.3.30 Policy Engine States

Table 8-80 lists the states used by the various state machines.

Table 8.157 Policy Engine States

State name	Reference
SenderResponseTimer	
<i>SRT_Stopped</i>	Section 8.3.3.1.1
<i>SRT_Running</i>	Section 8.3.3.1.1
<i>SRT_Expired</i>	Section 8.3.3.1.1
Source Port	
<i>PE_SRC_Startup</i>	Section 8.3.3.2.1
<i>PE_SRC_Discovery</i>	Section 8.3.3.2.2
<i>PE_SRC_Send_Capabilities</i>	Section 8.3.3.2.3
<i>PE_SRC_Negotiate_Capability</i>	Section 8.3.3.2.4
<i>PE_SRC_Transition_Supply</i>	Section 8.3.3.2.5
<i>PE_SRC_Ready</i>	Section 8.3.3.2.6
<i>PE_SRC_Disabled</i>	Section 8.3.3.2.7
<i>PE_SRC_Capability_Response</i>	Section 8.3.3.2.8
<i>PE_SRC_Hard_Reset</i>	Section 8.3.3.2.9
<i>PE_SRC_Hard_Reset_Received</i>	Section 8.3.3.2.10
<i>PE_SRC_Transition_to_default</i>	Section 8.3.3.2.11
<i>PE_SRC_Give_Source_Cap</i>	Section 8.3.3.2.15
<i>PE_SRC_Get_Sink_Cap</i>	Section 8.3.3.2.12
<i>PE_SRC_Wait_New_Capabilities</i>	Section 8.3.3.2.13
<i>PE_SRC_EPR_Keep_Alive</i>	Section 8.3.3.2.14
Sink Port	
<i>PE_SNK_Startup</i>	Section 8.3.3.3.1
<i>PE_SNK_Discovery</i>	Section 8.3.3.3.2
<i>PE_SNK_Wait_for_Capabilities</i>	Section 8.3.3.3.3
<i>PE_SNK_Evaluate_Capability</i>	Section 8.3.3.3.4
<i>PE_SNK_Select_Capability</i>	Section 8.3.3.3.5
<i>PE_SNK_Transition_Sink</i>	Section 8.3.3.3.6
<i>PE_SNK_Ready</i>	Section 8.3.3.3.7
<i>PE_SNK_Hard_Reset</i>	Section 8.3.3.3.8
<i>PE_SNK_Transition_to_default</i>	Section 8.3.3.3.9
<i>PE_SNK_Give_Sink_Cap</i>	Section 8.3.3.3.10
<i>PE_SNK_Get_Source_Cap</i>	Section 8.3.3.3.12
<i>PE_SNK_EPR_Keep_Alive</i>	Section 8.3.3.3.11
Soft Reset and Protocol Error	
Source Port Soft Reset	
<i>PE_SRC_Send_Soft_Reset</i>	Section 8.3.3.4.1.1
<i>PE_SRC_Soft_Reset</i>	Section 8.3.3.4.1.2
Sink Port Soft Reset	
<i>PE_SNK_Send_Soft_Reset</i>	Section 8.3.3.4.2.1
<i>PE_SNK_Soft_Reset</i>	Section 8.3.3.4.2.2
Data Reset	

DFP Data Reset	
<i>PE_DDR_Send_Data_Reset</i>	Section 8.3.3.5.1.1
<i>PE_DDR_Data_Reset_Received</i>	Section 8.3.3.5.1.2
<i>PE_DDR_Wait_For_VCONN_Off</i>	Section 8.3.3.5.1.3
<i>PE_DDR_Perform_Data_Reset</i>	Section 8.3.3.5.1.4
UFP Data Reset	
<i>PE_UDR_Send_Data_Reset</i>	Section 8.3.3.5.2.1
<i>PE_UDR_Data_Reset_Received</i>	Section 8.3.3.5.2.2
<i>PE_UDR_Turn_Off_VCONN</i>	Section 8.3.3.5.2.3
<i>PE_UDR_Send_Ps_Rdy</i>	Section 8.3.3.5.2.4
<i>PE_UDR_Wait_For_Data_Reset_Complete</i>	Section 8.3.3.5.2.5
Not Supported Message	
Source Port Not Supported	
<i>PE_SRC_Send_Not_Supported</i>	Section 8.3.3.6.1.1
<i>PE_SRC_Not_Supported_Received</i>	Section 8.3.3.6.1.2
<i>PE_SRC_Chunk_Received</i>	Section 8.3.3.6.1.3
Sink Port Not Supported	
<i>PE_SNK_Send_Not_Supported</i>	Section 8.3.3.6.2.1
<i>PE_SNK_Not_Supported_Received</i>	Section 8.3.3.6.2.2
<i>PE_SNK_Chunk_Received</i>	Section 8.3.3.6.2.3
Source Port Ping	
<i>PE_SRC_Ping</i>	Section 8.3.3.7.1
Source Alert	
Source Port Source Alert	
<i>PE_SRC_Send_Source_Alert</i>	Section 8.3.3.8.1.1
<i>PE_SRC_Wait_for_Get_Status</i>	Section 8.3.3.8.1.2
Sink Port Source Alert	
<i>PE_SNK_Source_Alert_Received</i>	Section 8.3.3.8.2.1
Sink Port Sink Alert	
<i>PE_SNK_Send_Sink_Alert</i>	Section 8.3.3.8.3.1
<i>PE_SNK_Wait_for_Get_Status</i>	Section 8.3.3.8.3.2
Source Port Sink Alert	
<i>PE_SRC_Sink_Alert_Received</i>	Section 8.3.3.8.4.1
Source/Sink Extended Capabilities	
Sink Port Get Source Capabilities Extended	
<i>PE_SNK_Get_Source_Cap_Ext</i>	Section 8.3.3.9.1.1
Source Port Give Source Capabilities Extended	
<i>PE_SRC_Give_Source_Cap_Ext</i> *	Section 8.3.3.9.2.1
Source Port Get Sink Capabilities Extended	
<i>PE_SRC_Get_Sink_Cap_Ext</i>	Section 8.3.3.9.3.1
Source Port Give Source Capabilities Extended	
<i>PE_SNK_Give_Sink_Cap_Ext</i> *	Section 8.3.3.9.4.1
Source Information	
Sink Port Get Source Information	

PE_SNK_Get_Source_Info	Section 8.3.3.10.1.1
Source Port Give Source Information	
PE_SRC_Give_Source_Info	Section 8.3.3.10.2.1
Status	
Get Status	
PE_Get_Status	Section 8.3.3.10.1.1
Give Status	
PE_Give_Status	Section 8.3.3.10.2.1
Sink Port Get PPS Status	
PE_SNK_Get_PPS_Status	Section 8.3.3.10.5.1
Source Port Give PPS Status	
PE_SRC_Give_PPS_Status	Section 8.3.3.10.6.1
Battery Capabilities	
Get Battery Capabilities	
PE_Get_Battery_Cap	Section 8.3.3.11.1.1
Give Battery Capabilities	
PE_Give_Battery_Cap	Section 8.3.3.11.2.1
Battery Status	
Get Battery Status	
PE_Get_Battery_Status	Section 8.3.3.12.1.1
Give Battery Status	
PE_Give_Battery_Status	Section 8.3.3.12.2.1
Manufacturer Information	
Get Manufacturer Information	
PE_Get_Manufacturer_Info	Section 8.3.3.13.1
Give Manufacturer Information	
PE_Give_Manufacturer_Info	Section 8.3.3.13.2
Country Codes and Information	
Get Country Codes	
PE_Get_Country_Codes	Section 8.3.3.14.1.1
Give Country Codes	
PE_Give_Country_Codes	Section 8.3.3.14.2.1
Get Country Information	
PE_Get_Country_Info	Section 8.3.3.14.3.1
Give Country Information	
PE_Give_Country_Info	Section 8.3.3.14.4.1
Revision	
Get Revision	
PE_Get_Revision	Section 8.3.3.15.1.1
Give Revision	
PE_Give_Revision	Section 8.3.3.15.2.1
Enter USB	

DFP Enter USB	
PE_DEU_Send_Enter_USB	Section 8.3.3.15.1.1
UFP Enter USB	
PE_UEU_Enter_USB_Received	Section 8.3.3.15.2.1
Security Request/Response	
Send Security Request	
PE_Send_Security_Request	Section 8.3.3.16.1
Send Security Response	
PE_Send_Security_Response	Section 8.3.3.16.2
Security Response Received	
PE_Security_Response_Received	Section 8.3.3.16.3
Firmware Update Request/Response	
Send Firmware Update Request	
PE_Send_Firmware_Update_Request	Section 8.3.3.17.1.1
Send Firmware Update Response	
PE_Send_Firmware_Update_Response	Section 8.3.3.17.2.1
Firmware Update Response Received	
PE_Firmware_Update_Response_Received	Section 8.3.3.17.3.1
Dual-Role Port	
DFP to UFP Data Role Swap	
PE_DRS_DFP_UFP_Evaluate_Swap	Section 8.3.3.18.1.2
PE_DRS_DFP_UFP_Accept_Swap	Section 8.3.3.18.1.3
PE_DRS_DFP_UFP_Change_to_UFP	Section 8.3.3.18.1.4
PE_DRS_DFP_UFP_Send_Swap	Section 8.3.3.18.1.5
PE_DRS_DFP_UFP_Reject_Swap	Section 8.3.3.18.1.6
UFP to DFP Data Role Swap	
PE_DRS_UFP_DFP_Evaluate_Swap	Section 8.3.3.18.2.2
PE_DRS_UFP_DFP_Accept_Swap	Section 8.3.3.18.2.3
PE_DRS_UFP_DFP_Change_to_DFP	Section 8.3.3.18.2.4
PE_DRS_UFP_DFP_Send_Swap	Section 8.3.3.18.2.5
PE_DRS_UFP_DFP_Reject_Swap	Section 8.3.3.18.2.6
Source to Sink Power Role Swap	
PE_PRS_SRC_SNK_Evaluate_Swap	Section 8.3.3.18.3.2
PE_PRS_SRC_SNK_Accept_Swap	Section 8.3.3.18.3.3
PE_PRS_SRC_SNK_Transition_to_off	Section 8.3.3.18.3.4
PE_PRS_SRC_SNK Assert_Rd	Section 8.3.3.18.3.5
PE_PRS_SRC_SNK_Wait_Source_on	Section 8.3.3.18.3.6
PE_PRS_SRC_SNK_Send_Swap	Section 8.3.3.18.3.7
PE_PRS_SRC_SNK_Reject_Swap	Section 8.3.3.18.3.8
Sink to Source Power Role Swap	
PE_PRS_SNK_SRC_Evaluate_Swap	Section 8.3.3.18.4.2
PE_PRS_SNK_SRC_Accept_Swap	Section 8.3.3.18.4.3
PE_PRS_SNK_SRC_Transition_to_off	Section 8.3.3.18.4.4
PE_PRS_SNK_SRC Assert_Rp	
PE_PRS_SNK_SRC_Source_on	Section 8.3.3.18.4.5

PE_PRS_SNK_SRC_Send_Swap	Section 8.3.3.18.4.7
PE_PRS_SNK_SRC_Reject_Swap	Section 8.3.3.18.4.8
Source to Sink Fast Role Swap	
PE_FRS_SRC_SNK_Evaluate_Swap	Section 8.3.3.18.5.2
PE_FRS_SRC_SNK_Accept_Swap	Section 8.3.3.18.5.3
PE_FRS_SRC_SNK_Transition_to_off	Section 8.3.3.18.5.4
PE_FRS_SRC_SNK Assert_Rd	Section 8.3.3.18.5.5
PE_FRS_SRC_SNK_Wait_Source_on	Section 8.3.3.18.5.6
Sink to Source Fast Role Swap	
PE_FRS_SNK_SRC_Start_AMS	Section 8.3.3.18.6.1
PE_FRS_SNK_SRC_Send_Swap	Section 8.3.3.18.6.2
PE_FRS_SNK_SRC_Transition_to_off	Section 8.3.3.18.6.3
PE_FRS_SNK_SRC_Vbus_Applied	Section 8.3.3.18.6.4
PE_FRS_SNK_SRC Assert_Rp	Section 8.3.3.18.6.5
PE_FRS_SNK_SRC_Source_on	Section 8.3.3.18.6.6
Dual-Role Source Port Get Source Capabilities	
PE_DR_SRC_Get_Source_Cap	Section 8.3.3.18.7.1
Dual-Role Source Port Give Sink Capabilities	
PE_DR_SRC_Give_Sink_Cap	Section 8.3.3.18.8.1
Dual-Role Sink Port Get Sink Capabilities	
PE_DR_SNK_Get_Sink_Cap	Section 8.3.3.18.9.1
Dual-Role Sink Port Give Source Capabilities	
PE_DR_SNK_Give_Source_Cap	Section 8.3.3.18.10.1
Dual-Role Source Port Get Source Capabilities Extended	
PE_DR_SRC_Get_Source_Cap_Ext	Section 8.3.3.18.11.1
Dual-Role Sink Port Give Source Capabilities Extended	
PE_DR_SNK_Give_Source_Cap_Ext	Section 8.3.3.18.12.1
Dual-Role Sink Port Get Sink Capabilities Extended	
PE_DR_SNK_Get_Sink_Cap_Ext	Section 8.3.3.19.13.1
Dual-Role Source Port Give Sink Capabilities Extended	
PE_DR_SRC_Give_Sink_Cap_Ext	Section 8.3.3.19.14.1
Dual-Role Source Port Get Source Information	
PE_DR_SRC_Get_Source_Info	Section 8.3.3.20.15.1
Dual-Role Sink Port Give Source Information	
PE_DR_SNK_Give_Source_Info	Section 8.3.3.20.16.1
USB Type-C® VCONN Swap	
PE_VCS_Send_Swap	Section 8.3.3.19.1
PE_VCS_Evaluate_Swap	Section 8.3.3.19.2
PE_VCS_Accept_Swap	Section 8.3.3.19.3
PE_VCS_Reject_Swap	Section 8.3.3.19.4
PE_VCS_Wait_For_VCONN	Section 8.3.3.19.5
PE_VCS_Turn_Off_VCONN	Section 8.3.3.19.6
PE_VCS_Turn_On_VCONN	Section 8.3.3.19.7
PE_VCS_Send_Ps_Rdy	Section 8.3.3.19.8
PE_VCS_Force_VCONN	Section 8.3.3.19.9

Initiator Structured VDM	
Initiator to Port Structured VDM Discover Identity	
PE_INIT_PORT_VDM_Identity_Request	Section 8.3.3.20.1.1
PE_INIT_PORT_VDM_Identity_ACKed	Section 8.3.3.20.1.2
PE_INIT_PORT_VDM_Identity_NAKed	Section 8.3.3.20.1.3
Initiator Structured VDM Discover SVIDs	
PE_INIT_VDM_SVIDs_Request	Section 8.3.3.20.2.1
PE_INIT_VDM_SVIDs_ACKed	Section 8.3.3.20.2.2
PE_INIT_VDM_SVIDs_NAKed	Section 8.3.3.20.2.3
Initiator Structured VDM Discover Modes	
PE_INIT_VDM_Modes_Request	Section 8.3.3.20.3.1
PE_INIT_VDM_Modes_ACKed	Section 8.3.3.20.3.2
PE_INIT_VDM_Modes_NAKed	Section 8.3.3.20.3.3
Initiator Structured VDM Attention	
PE_INIT_VDM_Attention_Request	Section 8.3.3.20.4.1
Responder Structured VDM	
Responder Structured VDM Discovery Identity	
PE_RESP_VDM_Get_Identity	Section 8.3.3.21.1.1
PE_RESP_VDM_Send_Identity	Section 8.3.3.21.1.2
PE_RESP_VDM_Get_Identity_NAK	Section 8.3.3.21.1.3
Responder Structured VDM Discovery SVIDs	
PE_RESP_VDM_Get_SVIDs	Section 8.3.3.21.2.1
PE_RESP_VDM_Send_SVIDs	Section 8.3.3.21.2.2
PE_RESP_VDM_Get_SVIDs_NAK	Section 8.3.3.21.2.3
Responder Structured VDM Discovery Modes	
PE_RESP_VDM_Get_Modes	Section 8.3.3.21.3.1
PE_RESP_VDM_Send_Modes	Section 8.3.3.21.3.2
PE_RESP_VDM_Get_Modes_NAK	Section 8.3.3.21.3.3
Receiving a Structured VDM Attention	
PE_RCV_VDM_Attention_Request	Section 8.3.3.21.4.1
DFP Structured VDM	
DFP Structured VDM Mode Entry	
PE_DFP_VDM_Mode_Entry_Request	Section 8.3.3.22.1.1
PE_DFP_VDM_Mode_Entry_ACKed	Section 8.3.3.22.1.2
PE_DFP_VDM_Mode_Entry_NAKed	Section 8.3.3.22.1.3
DFP Structured VDM Mode Exit	
PE_DFP_VDM_Mode_Exit_Request	Section 8.3.3.22.2.1
PE_DFP_VDM_Mode_Exit_ACKed	Section 8.3.3.22.2.2
UFP Structure VDM	
UFP Structured VDM Enter Mode	
PE_UFP_VDM_Evaluate_Mode_Entry	Section 8.3.3.23.1.1
PE_UFP_VDM_Mode_Entry_ACK	Section 8.3.3.23.1.2
PE_UFP_VDM_Mode_Entry_NAK	Section 8.3.3.23.1.3
UFP Structured VDM Exit Mode	

PE_UFP_VDM_Mode_Exit	Section 8.3.3.23.2.1
PE_UFP_VDM_Mode_Exit_ACK	Section 8.3.3.23.2.2
PE_UFP_VDM_Mode_Exit_NAK	Section 8.3.3.23.2.3
Cable Plug Specific	
Cable Ready	
PE_CBL_Ready	Section 8.3.3.24.1.1
Mode Entry	
PE_CBL_Evaluate_Mode_Entry	Section 8.3.3.24.4.1.1
PE_CBL_Mode_Entry_ACK	Section 8.3.3.24.4.1.2
PE_CBL_Mode_Entry_NAK	Section 8.3.3.24.4.1.3
Mode Exit	
PE_CBL_Mode_Exit	Section 8.3.3.24.4.2.1
PE_CBL_Mode_Exit_ACK	Section 8.3.3.24.4.2.2
PE_CBL_Mode_Exit_NAK	Section 8.3.3.24.4.2.3
Cable Soft Reset	
PE_CBL_Soft_Reset	Section 8.3.3.24.2.1.1
Cable Hard Reset	
PE_CBL_Hard_Reset	Section 8.3.3.24.2.2.1
DFP/VCONN Source Soft Reset or Cable Reset	
PE_DFP_VCS_CBL_Send_Soft_Reset	Section 8.3.3.24.2.3.1
PE_DFP_VCS_CBL_Send_Cable_Reset	Section 8.3.3.24.2.3.2
UFP/VCONN Source Soft Reset or Cable Reset	
PE_UFP_VCS_CBL_Send_Soft_Reset	Section 8.3.3.24.2.4.1
Source Startup Structured VDM Discover Identity	
PE_SRC_VDM_Identity_Request	Section 8.3.3.24.3.1
PE_SRC_VDM_Identity_ACKed	Section 8.3.3.24.3.2
PE_SRC_VDM_Identity_NAKed	Section 8.3.3.24.3.3
EPR Mode	
Source EPR Mode Entry	
PE_SRC_Evaluate_EPR_Mode_Entry	Section 8.3.3.25.1.1
PE_SRC_EPR_Mode_Entry_Ack	Section 8.3.3.25.1.2
PE_SRC_EPR_Mode_Discover_Cable	Section 8.3.3.25.1.3
PE_SRC_EPR_Mode_Evaluate_Cable_EPR	Section 8.3.3.25.1.4
PE_SRC_EPR_Mode_Entry_Succeeded	Section 8.3.3.25.1.5
PE_SRC_EPR_Mode_Entry_Failed	Section 8.3.3.25.1.6
Sink EPR Mode Entry	
PE_SNK_Send_EPR_Mode_Entry	Section 8.3.3.25.2.1
PE_SNK_EPR_Mode_Wait_For_Response	Section 8.3.3.25.2.2
Source EPR Mode Exit	
PE_SRC_Send_EPR_Mode_Exit	Section 8.3.3.25.3.1
PE_SRC_EPR_Mode_Exit_Received	Section 8.3.3.25.3.2
Sink EPR Mode Exit	
PE_SNK_Send_EPR_Mode_Exit	Section 8.3.3.25.4.1
PE_SNK_EPR_Mode_Exit_Received	Section 8.3.3.25.4.2
BIST	

	BIST Carrier Mode
<u>PE_BIST_Carrier_Mode</u>	<u>Section 8.3.3.26.1.1</u>
	✖ BIST Carrier Mode
<u>PE_BIST_Test_Mode</u>	<u>Section 8.3.3.27.2</u> ✖
	BIST Shared Capacity Test Mode
<u>PE_BIST_Shared_Capacity_Test_Mode</u>	<u>Section 8.3.3.27.3</u>
	USB Type-C® referenced states
<u>ErrorRecovery</u>	<u>Section 8.3.3.27.1</u>