Configuring Ethernet Virtual Connection Bridge Domain

Ethernet virtual connection (EVC) infrastructure is a Layer 2 platform-independent bridging architecture that supports Ethernet services. This chapter provides procedures for configuring EVC Bridge Domain (BD) and the features it supports on the Cisco Integrated Services Routers (ISR) G2.

- Configuring EVCs on Cisco ISR G2 Router, page 61
- Ethernet Data Plane Loopback, page 64
- Connectivity Fault Management (CFM) over EVC BD, page 68
- Support for Y.1731 Performance Monitoring for EVC BD, page 87
- Support for Switch Virtual Interfaces (SVI) on ISR G2 Metro Ethernet BD, page 90
- EVC Quality of Service (QoS), page 92

Configuring EVCs on Cisco ISR G2 Router

Configuring an EFP and a BD on the Cisco ISR G2 Router

Configuring a service instance on a Layer 2 port creates an EFP on which you can configure EVC features.

Note

- You cannot use the same VLAN ID for encapsulating on a Layer 3 sub-interface and an EFP (service instance) on a WAN interface.
- If there is a sub-interface and service-instance both configured on a WAN interface for untagged traffic, then the traffic will always go to the main interface and the service-instance with untagged traffic will not work.

Perform this task to configure an EFP.

SUMMARY STEPS

Step 1 **enable**

Step 2 **configure terminal**

Step 3 **interface** *type number*

Step 4 **service instance** *id* **ethernet**

Step 5 **encapsulation** *encapsulation-type vlan-id*

Step 6 **rewrite ingress tag translate 1-to-1 dot1q** *vlan-id* **symmetric**

Step 7 **bridge-domain** *bridge-id*

Step 8 **end**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface gigabitethernet 0/1`	Enters interface configuration mode. • The example shows how to configure Gigabit Ethernet interface 0/1 and enter interface configuration mode.
Step 4	**service instance** *id* **ethernet** **Example:** `Router(config-if)# service instance 1 ethernet`	Configures an Ethernet service instance on an interface and enters Ethernet service configuration mode. • The example shows how to configure Ethernet service instance 1.
Step 5	**encapsulation** *encapsulation-type vlan-id* **Example:** `Router(config-if-srv)# encapsulation dot1q 1`	Defines the encapsulation type. • The example shows how to define dot1q as the encapsulation type.
Step 6	**rewrite ingress tag translate 1-to-1 dot1q** *vlan-id* **symmetric** **Example:** `Router(config-if-srv)# rewrite ingress tag translate 1-to-1 dot1q 1 symmetric`	(Optional) Specifies the encapsulation adjustment to be performed on a frame ingressing a service instance. • The example shows how to specify translating a single tag defined by the **encapsulation** command to a single tag defined in the **rewrite ingress tag** command with reciprocal adjustment to be done in the egress direction.

	Command	Purpose
Step 7	**bridge-domain** *bridge-id* **Example:** `Router(config-if-srv)# bridge-domain 1`	Configures the bridge domain. • The example shows how to configure bridge domain 1.
Step 8	**end** **Example:** `router(config-if-srv)# end`	Returns to privileged EXEC mode.

Configuration Examples for EVCs on the Cisco ISR G2 Router

When a WAN interface is configured with both an EFP and a subinterface, and the dot1q encapsulation with the same VLAN ID is used, the traffic on the subinterface gets a higher priority than the traffic on an EFP.

Note the following configuration order before you configure EVC:

Order 1: If you configure the subinterface with the same VLAN ID first, then the configuration of EFP using the same VLAN ID is blocked as shown below:

```
router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
router(config)#int gi0/0
irouter(config-if)# service instance 2 ethernet evc1
router(config-if-srv)# encapsulation dot1q 102
Invalid configuration on ServInst 2(Gi0/0). The VLAN ID (102) has already been configured
on interface GigabitEthernet0/0.102
```

Order 2: If you configure EFP first using the same VLAN ID, then you can still configure the subinterface using the same VLAN ID. However, traffic will flow on the subinterface with higher priority and not on the EFP.

Configuring an EFP and a subinterface using the same VLAN ID for dot1q encapsulation is allowed and configurable as show in order 2. However, the use of an EFP and subinterface is mutually exclusive. There will not be any traffic through the EFP. Traffic only goes through the subinterface because untagged packets have high priority than tagged packets.

Example Configuring EFPs on a Gigabit Ethernet Interface

```
interface GigabitEthernet0/1
 no ip address
 negotiation auto
 service instance 1 ethernet
 encapsulation dot1q 201
 rewrite ingress tag translate 1-to-1 dot1q 300 symmetric
 bridge-domain 1
 !
 service instance 2 ethernet
 encapsulation default
 bridge-domain 1
 !
 service instance 3 ethernet
 encapsulation priority-tagged
 bridge-domain 2
 !
```

Ethernet Data Plane Loopback

The Ethernet Data Plane Loopback feature allows you to test services and throughput of an Ethernet port or a device using a test generator. You can verify the maximum rate of frame transmission with no frame loss. This feature allows bidirectional throughput measurement, and on-demand or out-of-service (intrusive) operation during service turn-ups. This feature is used for testing during service turn-ups and troubleshooting of services after a turn-up.

If you need to test a service while it is live, you do this without disrupting any of the live data traffic. To achieve this, you use test traffic that differs from live data traffic. For example, the traffic from a test generator contain the source MAC address of the test generator, or test traffic is assigned a particular Class of Service (Cos). Irrespective of the method used, the device looping back the traffic must be able to filter out the test traffic and leave the data traffic untouched.

Note Configuring Ethernet Data Plane Loopback on a device does not indicate the start of an actual session.

Features Supported for Ethernet Data Plane Loopback

- Locally-enabled Ethernet Data Plane Loopback on all Ethernet interface types, such as physical and bundle interfaces and sub-interfaces and Pseudowire Head End (PWHE) interfaces.
- In the case of Layer 2 and Layer 3 interfaces, only external loopback is supported. External loopback is the type of loopback where all traffic received on the ingress interface is blindly sent out of the egress interface.
- When a Bundle interface is placed into loopback, traffic on all bundle link members are looped back.
- MAC address must always be swapped on looped-back traffic.
- Supports dropping of packets received in the non-loopback direction.
- Allows the application of multiple filters to loopback only a subset of traffic received by an interface and only drop the corresponding reverse-direction traffic.
- Provides an option to specify a time period after which the loopback is automatically terminated.

Restrictions of Ethernet Data Plane Loopback

- EVC xconnect is not supported.
- Only single VLAN is supported as the filtering options, but the vlan-list/vlan range is not supported.
- Maximum of 10 active sessions is only supported.

Configuring Ethernet Data Plane Loopback

Perform this task to configure Ethernet Data Plane Loopback.

Note Configuring or permitting Ethernet Data Plane Loopback is not the same as starting an actual loopback session.

SUMMARY STEPS

Step 1 **configure**

Step 2 **interface [GigabitEthernet |]** *interface-path-id*

Step 3 **ethernet loopback permit {external | internal}**

Step 4 **end**

or

commit

DETAILED STEPS

	Command	Purpose
Step 1	**configure** **Example:** `Router# configure`	Enters global configuration mode.
Step 2	**interface [GigabitEthernet |]** *interface-path-id* **Example:** `router(config)# interface 0/1`	Enters interface configuration mode and specifies the Ethernet interface name and notation *rack/slot/module/port*. **Note** The example indicates an 8-port 10-Gigabit Ethernet interface in modular services card slot 1.

	Command	Purpose
Step 3	**ethernet loopback permit {external | internal}** **Example:** `Router(config-if-srv)# ethernet loopback permit external`	Configures ethernet loopback externally or internally on an interface. External loopback allows loopback of traffic from wire. Internal loopback allows loopback of traffic from the bridge domain.
Step 4	**end** or **commit** **Example:** `router(config-if-srv)# commit`	Saves configuration changes. • When you issue the end command, the system prompts you to commit changes: Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]: – Entering yes saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. – Entering no exits the configuration session and returns the router to EXEC mode without committing the configuration changes. – Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the commit command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for Ethernet Data Plane Loopback

This example shows how to configure Ethernet Data Plane Loopback:

```
Router# configure
Router(config)# interface GigabitEthernet 0/1
Router((config-if-srv)# ethernet loopback permit external
```

This example shows how to start an Ethernet Data Plane Loopback:

```
Router# ethernet loopback start local interface gigabitEthernet 0/1
external
[source mac-address <addr>]
[destination mac-address <addr>]
[ether-type <etype>]
[{dot1q <vlan-ids> [second-dot1q <vlan-ids>] |
dot1ad <vlan-ids> [dot1q <vlan-ids>]}]
[cos <cos>]
[llc-oui <oui>]
[timeout {<length> | none}]
```

This example shows how to stop an Data Plane Loopback:

```
Router# ethernet loopback stop local interface <name> id <id>
```

This example shows how to extend an Ethernet Data Plane Loopback:

```
router# ethernet loopback extend local interface <name> id <id> length
```

```
<length>
```

Verifying the Ethernet Data Plane Loopback Configuration

Use the **show ethernet loopback permitted** command to display all the permitted interfaces which run Ethernet Data Plane Loopback sessions:

```
router# show ethernet loopback permitted

Interface Direction
--------------------------------------------------------------------------------
GigabitEthernet0/0/0/0 External
GigabitEthernet0/0/0/1.100 Internal
GigabitEthernet 0/1.200 External, Internal
```

Use the **show ethernet loopback active** command to view active sessions:

```
Router# show ethernet loopback active interface GigabitEthernet 0/1.200

Local: GigabitEthernet0/1.200, ID 1
--------------------------------------------------------------------------------
Direction: Internal
Time out: 2 hours
Time left: 00:01:17
Status: Active
Filters:
Dot1ad: 100-200
Dot1q: Any
Source MAC Address: aaaa.bbbb.cccc
Destination MAC Address: Any
Ethertype: 0x8902
Class of Service: Any
LLC-OUI: Any
Local: GigabitEthernet0/1.200, ID 2
--------------------------------------------------------------------------------
Direction: External
Time out: 10 minutes
Time left: 00:00:00
Status: Stopping
Filters:
Dot1q: 500
Second-dot1q: 200
Source MAC Address: Any
Destination MAC Address: Any
Ethertype: Any
Class of Service: 4
LLC-OUI: Any
```

For each loopback session listed, this information is displayed:

- Header containing the Interface name and session ID, which uniquely identify the local loopback session,
- Direction which specifies the direction of the loopback,
- Time out – the time out period specified when the loopback was started,
- Time left – the amount of time left until the loopback session is automatically stopped,
- Status – the status of the loopback session,
- Filters – details of the filters specified when the loopback session was started. Similar to the start CLI, only the filters supported by the platform are displayed.

Connectivity Fault Management (CFM) over EVC BD

IEEE CFM is an end-to-end per-service Ethernet-layer Operations, Administration, and Maintenance (OAM) protocol. CFM includes proactive connectivity monitoring, fault verification, and fault isolation for large Ethernet metropolitan-area networks (MANs) and WANs.

CFM over EVC BD (Up mep) and CFM over EVC BD (Down mep) features are supported on CFM over EVC BD.

CFM over Xconnect (Up mep) and CFM over Xconnect (Down mep) features are not supported on CFM over EVC BD.

The benefits of Ethernet CFM are:

- End-to-end service-level OAM technology
- Reduced operating expense for service provider Ethernet networks
- Competitive advantage for service providers

Note This feature is supported only if you have purchased the *appxk9* licensing package. CFM over EVC BD is available only on the Cisco 890 series ISR and ISRG2 platforms. For more information about managing software activation licenses on the Cisco ISR and Cisco ISR G2 platforms, see http://www.cisco.com/en/US/docs/routers/access/sw_activation/SA_on_ISR.html.

Restrictions for Configuring Ethernet CFM

- A specific domain must be configured. If it is not, an error message is displayed.
- Multiple domains (different domain names) having the same maintenance level can be configured. However, associating a single domain name with multiple maintenance levels is not permitted.

Configuring Ethernet CFM

- Provisioning the Network (CE-A), page 68
- Provisioning the Network (CE-B), page 70
- Provisioning Service (CE-A), page 73
- Provisioning Service (CE-B), page 76
- Configuring and Enabling the Cross-Check Function (CE-A), page 78
- Configuring and Enabling the Cross-Check Function (CE-B), page 80
- Configuration Examples for Configuring Ethernet CFM for the Cisco ISR G2 Routers, page 81

Provisioning the Network (CE-A)

Complete these steps to configure provisioning the network (CE-A):

SUMMARY STEPS

Step 1 **enable**

Step 2 **configure terminal**

Step 3 **ethernet cfm domain** *domain-name* **level** *level-id*

Step 4 **mep archive-hold-time** *minutes*

Step 5 **exit**

Step 6 **ethernet cfm global**

Step 7 **ethernet cfm ieee**

Step 8 **ethernet cfm traceroute cache**

Step 9 **ethernet cfm traceroute cache size** *entries*

Step 10 **ethernet cfm traceroute cache hold-time** *minutes*

Step 11 **snmp-server enable traps ethernet cfm cc [mep-up] [mep-down] [config] [loop] [cross-connect]**

Step 12 **snmp-server enable traps ethernet cfm crosscheck [mep-unknown] [mep-missing] [service-up]**

Step 13 **end**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables the privileged EXEC mode. Enter your password when prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters the global configuration mode.
Step 3	**ethernet cfm domain** *domain-name* **level** *level-id* **Example:** `Router(config)# ethernet cfm domain Customer level 7`	Defines a CFM maintenance domain at a particular maintenance level and enters Ethernet CFM configuration mode.
Step 4	**mep archive-hold-time** *minutes* **Example:** `Router(config-ecfm)# mep archive-hold-time 60`	Sets the amount of time that data from a missing MEP is kept in the continuity check database or that entries are held in the error database before they are purged.
Step 5	**exit** **Example:** `Router(config-ecfm)# exit`	Returns the device to global configuration mode.
Step 6	**ethernet cfm global** **Example:** `Router(config)# ethernet cfm global`	Enables CFM processing globally on the device.

	Command	Purpose
Step 7	**ethernet cfm ieee** **Example:** `Router(config)# ethernet cfm ieee`	Enables the CFM IEEE version of CFM. • This command is automatically issued when the ethernet cfm global command is issued.
Step 8	**ethernet cfm traceroute cache** **Example:** `Router(config)# ethernet cfm traceroute cache`	Enables caching of CFM data learned through traceroute messages.
Step 9	**ethernet cfm traceroute cache size** *entries* **Example:** `Router(config)# ethernet cfm traceroute cache size 200`	Sets the maximum size for the CFM traceroute cache table.
Step 10	**ethernet cfm traceroute cache hold-time** *minutes* **Example:** `Router(config)# ethernet cfm traceroute cache hold-time 60`	Sets the amount of time that CFM traceroute cache entries are retained.
Step 11	**snmp-server enable traps ethernet cfm cc [mep-up] [mep-down] [config] [loop] [cross-connect]** **Example:** `Router(config)# snmp-server enable traps ethernet cfm cc mep-up mep-down config loop cross-connect`	Enables SNMP trap generation for Ethernet CFM continuity check events.
Step 12	**snmp-server enable traps ethernet cfm crosscheck [mep-unknown] [mep-missing] [service-up]** **Example:** `Router(config)# snmp-server enable traps ethernet cfm crosscheck mep-unknown`	Enables SNMP trap generation for Ethernet CFM continuity check events in relation to the cross-check operation between statically configured MEPs and those learned via CCMs.
Step 13	**end** **Example:** `Router(config)# end`	Returns the router to the privileged EXEC mode.

Provisioning the Network (CE-B)

Complete these steps to configure provisioning the network (CE-B):

SUMMARY STEPS

Step 1 **enable**

Step 2 **configure terminal**

Step 3 **ethernet cfm domain** *domain-name* **level** *level-id* **[direction outward]**

Step 4 **mep archive-hold-time** *minutes*

Step 5 **exit**

Step 6 **ethernet cfm global**

Step 7 **ethernet cfm ieee**

Step 8 **ethernet cfm traceroute cache**

Step 9 **ethernet cfm traceroute cache size** *entries*

Step 10 **ethernet cfm traceroute cache hold-time** *minutes*

Step 11 **snmp-server enable traps ethernet cfm cc [mep-up] [mep-down] [config] [loop] [cross-connect]**

Step 12 **snmp-server enable traps ethernet cfm crosscheck [mep-unknown] [mep-missing] [service-up]**

Step 13 **end**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables the privileged EXEC mode. Enter your password when prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters the global configuration mode.
Step 3	**ethernet cfm domain** *domain-name* **level** *level-id* **[direction outward]** **Example:** `Router(config)# ethernet cfm domain Customer level 7 direction outward`	Defines an outward CFM maintenance domain at a particular maintenance level and enters Ethernet CFM configuration mode.
Step 4	**mep archive-hold-time** *minutes* **Example:** `Router(config-ecfm)# mep archive-hold-time 60`	Sets the amount of time that data from a missing MEP is kept in the continuity check database or that entries are held in the error database before they are purged.
Step 5	**exit** **Example:** `Router(config-ecfm)# exit`	Returns the device to global configuration mode.

	Command	Purpose
Step 6	**ethernet cfm global** **Example:** `Router(config)# ethernet cfm global`	Enables CFM processing globally on the device.
Step 7	**ethernet cfm ieee** **Example:** `Router(config)# ethernet cfm ieee`	Enables the CFM IEEE version of CFM. • This command is automatically issued when the ethernet cfm global command is issued.
Step 8	**ethernet cfm traceroute cache** **Example:** `Router(config)# ethernet cfm traceroute cache`	Enables caching of CFM data learned through traceroute messages.
Step 9	**ethernet cfm traceroute cache size** *entries* **Example:** `Router(config)# ethernet cfm traceroute cache size 200`	Sets the maximum size for the CFM traceroute cache table.
Step 10	**ethernet cfm traceroute cache hold-time** *minutes* **Example:** `Router(config)# ethernet cfm traceroute cache hold-time 60`	Sets the amount of time that CFM traceroute cache entries are retained.
Step 11	**snmp-server enable traps ethernet cfm cc [mep-up] [mep-down] [config] [loop] [cross-connect]** **Example:** `Router(config)# snmp-server enable traps ethernet cfm cc mep-up mep-down config loop cross-connect`	Enables SNMP trap generation for Ethernet CFM continuity check events.
Step 12	**snmp-server enable traps ethernet cfm crosscheck [mep-unknown] [mep-missing] [service-up]** **Example:** `Router(config)# snmp-server enable traps ethernet cfm crosscheck mep-unknown`	Enables SNMP trap generation for Ethernet CFM continuity check events in relation to the cross-check operation between statically configured MEPs and those learned via CCMs.
Step 13	**end** **Example:** `Router(config)# end`	Returns the router to the privileged EXEC mode.

Provisioning Service (CE-A)

Perform this task to set up service for Ethernet CFM. Optionally, when this task is completed, you may configure and enable the cross-check function. To perform this optional task, see "Configuring and Enabling the Cross-Check Function (CE-A)".

SUMMARY STEPS

Step 1 **enable**

Step 2 **configure terminal**

Step 3 **ethernet cfm domain** *domain-name* **level** *level-id*

Step 4 **service** {*ma-name* | *ma-num* | **vlan-id** *vlan-id* | **vpn-id** *vpn-id*} [**port** | **vlan** *vlan-id* [**direction down**]]

Step 5 **continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**]

Step 6 **continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**]

Step 7 **continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**]

Step 8 **exit**

Step 9 **mep archive-hold-time** *minutes*

Step 10 **exit**

Step 11 **ethernet cfm global**

Step 12 **ethernet cfm ieee**

Step 13 **ethernet cfm traceroute cache**

Step 14 **ethernet cfm traceroute cache size** *entries*

Step 15 **ethernet cfm traceroute cache hold-time** *minutes*

Step 16 **interface** *type number*

Step 17 **ethernet cfm mep domain** *domain-name* **mpid** *mpid* **{port | vlan** *vlan-id***}**

Step 18 **ethernet cfm mep domain** *domain-name* **mpid** *mpid* **{port | vlan** *vlan-id***}**

Step 19 **end**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables the privileged EXEC mode. Enter your password when prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters the global configuration mode.

	Command	Purpose
Step 3	**ethernet cfm domain** *domain-name* **level** *level-id* **Example:** `Router(config)# ethernet cfm domain Customer level 7`	Defines a CFM maintenance domain at a particular maintenance level and enters Ethernet CFM configuration mode.
Step 4	**service** {*ma-name* | *ma-num* | **vlan-id** *vlan-id* | **vpn-id** *vpn-id*} [**port** | **vlan** *vlan-id* [**direction down**]] **Example:** `Router(config-ecfm)# service Customer1 vlan 101 direction down`	Configures an MA within a maintenance domain and enters CFM service configuration mode. • If a service is already configured and you configure a new MA name and also specify the **direction down** keyword, a second service is added that maps to the same VLAN. If you configure a new MA name and do not specify the **direction down** keyword, the service is renamed to the new MA name.
Step 5	**continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**] **Example:** `Router(config-ecfm-srv)# continuity-check`	Enables the transmission of CCMs.
Step 6	**continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**] **Example:** `Router(config-ecfm-srv)# continuity-check interval 10`	Configures the time period between CCM transmissions. • The values supported are platform dependent.
Step 7	**continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**] **Example:** `Router(config-ecfm-srv)# continuity-check loss-threshold 10`	Sets the number of CCMs that should be missed before declaring that a remote MEP is down.
Step 8	**exit** **Example:** `Router(config-ecfm-srv)# exit`	Returns the device to Ethernet CFM configuration mode.
Step 9	**mep archive-hold-time** *minutes* **Example:** `Router(config-ecfm)# mep archive-hold-time 60`	Sets the amount of time that data from a missing MEP is kept in the continuity check database or that entries are held in the error database before they are purged.
Step 10	**exit** **Example:** `Router(config-ecfm)# exit`	Returns the device to global configuration mode.

	Command	Purpose
Step 11	**ethernet cfm global** **Example:** `Router(config)# ethernet cfm global`	Enables CFM processing globally on the device.
Step 12	**ethernet cfm ieee** **Example:** `Router(config)# ethernet cfm ieee`	Enables the CFM IEEE version of CFM. • This command is automatically issued when the ethernet cfm global command is issued.
Step 13	**ethernet cfm traceroute cache** **Example:** `Router(config)# ethernet cfm traceroute cache`	Enables caching of CFM data learned through traceroute messages.
Step 14	**ethernet cfm traceroute cache size** *entries* **Example:** `Router(config)# ethernet cfm traceroute cache size 200`	Sets the maximum size for the CFM traceroute cache table.
Step 15	**ethernet cfm traceroute cache hold-time** *minutes* **Example:** `Router(config)# ethernet cfm traceroute cache hold-time 60`	Sets the amount of time that CFM traceroute cache entries are retained.
Step 16	**interface** *type number* **Example:** `Router(config)# interface ethernet 0/3`	Specifies an interface and enters interface configuration mode.
Step 17	**ethernet cfm mep domain** *domain-name* **mpid** *mpid* **{port| vlan** *vlan-id***}** **Example:** `Router(config-if)# ethernet cfm mep domain Customer mpid 701 vlan 100`	Sets a port as internal to a maintenance domain and defines it as a MEP.
Step 18	**ethernet cfm mep domain** *domain-name* **mpid** *mpid* **{port| vlan** *vlan-id***}** **Example:** `Router(config-if)# ethernet cfm mep domain Customer mpid 701 vlan 100`	Sets a port as internal to a maintenance domain and defines it as a MEP.

Step 19	**end**
	
	Example:
	`Router(config-if)# end`
	`Router#`
		Returns the router to the privileged EXEC mode.

Provisioning Service (CE-B)

SUMMARY STEPS

Step 1 **enable**

Step 2 **configure terminal**

Step 3 **ethernet cfm domain** *domain-name* **level** *level-id* **[direction outward]**

Step 4 **mep archive-hold-time** *minutes*

Step 5 **service** {*ma-name* | *ma-num* | **vlan-id** *vlan-id* | **vpn-id** *vpn-id*} [**port** | **vlan** *vlan-id* [**direction down**]]

Step 6 **continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**]

Step 7 **continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**]

Step 8 **continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**]

Step 9 **exit**

Step 10 **exit**

Step 11 **ethernet cfm global**

Step 12 **ethernet cfm ieee**

Step 13 **ethernet cfm traceroute cache**

Step 14 **ethernet cfm traceroute cache size** *entries*

Step 15 **ethernet cfm traceroute cache hold-time** *minutes*

Step 16 **interface** *type number*

Step 17 **ethernet cfm mep level** *level-id* **[inward | outward domain** *domain-name*] **mpid** *id* **vlan{any |** *vlan-id* | *, vlan-id* | *vlan-id - vlan-id* | *vlan-id - vlan-id***}**

Step 18 **end**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables the privileged EXEC mode. Enter your password when prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters the global configuration mode.
Step 3	**ethernet cfm domain** *domain-name* **level** *level-id* **[direction outward]** **Example:** `Router(config)# ethernet cfm domain Customer level 7 direction outward`	Defines a CFM maintenance domain at a specified level and enters Ethernet CFM configuration mode.

	Command	Purpose
Step 4	**mep archive-hold-time** *minutes* **Example:** `Router(config-ecfm)# mep archive-hold-time 60`	Sets the amount of time that data from a missing MEP is kept in the continuity check database or that entries are held in the error database before they are purged.
Step 5	**service** {*ma-name* | *ma-num* | **vlan-id** *vlan-id* | **vpn-id** *vpn-id*} [**port** | **vlan** *vlan-id* [**direction down**]] **Example:** `Router(config-ecfm)# service Customer1 vlan 101 direction down`	Configures an MA within a maintenance domain and enters CFM service configuration mode. • If a service is already configured and you configure a new MA name and also specify the **direction down** keyword, a second service is added that maps to the same VLAN. If you configure a new MA name and do not specify the **direction down** keyword, the service is renamed to the new MA name.
Step 6	**continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**] **Example:** `Router(config-ecfm-srv)# continuity-check`	Enables the transmission of CCMs.
Step 7	**continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**] **Example:** `Router(config-ecfm-srv)# continuity-check interval 10`	Configures the time period between CCM transmissions. • The values supported are platform dependent.
Step 8	**continuity-check** [**interval** *time* | **loss-threshold** *threshold* | **static rmep**] **Example:** `Router(config-ecfm-srv)# continuity-check loss-threshold 10`	Sets the number of CCMs that should be missed before declaring that a remote MEP is down.
Step 9	**exit** **Example:** `Router(config-ecfm-srv)# exit`	Returns the device to Ethernet CFM configuration mode.
Step 10	**exit** **Example:** `Router(config-ecfm)# exit`	Returns the device to global configuration mode.
Step 11	**ethernet cfm global** **Example:** `Router(config)# ethernet cfm global`	Enables CFM processing globally on the device.

	Command	Purpose
Step 12	**ethernet cfm ieee** **Example:** `Router(config)# ethernet cfm ieee`	Enables the CFM IEEE version of CFM. • This command is automatically issued when the **ethernet cfm global** command is issued.
Step 13	**ethernet cfm traceroute cache** **Example:** `Router(config)# ethernet cfm traceroute cache`	Enables caching of CFM data learned through traceroute messages.
Step 14	**ethernet cfm traceroute cache size** *entries* **Example:** `Router(config)# ethernet cfm traceroute cache size 200`	Sets the maximum size for the CFM traceroute cache table.
Step 15	**ethernet cfm traceroute cache hold-time** *minutes* **Example:** `Router(config)# ethernet cfm traceroute cache hold-time 60`	Sets the amount of time that CFM traceroute cache entries are retained.
Step 16	**interface** *type number* **Example:** `Router(config)# interface ethernet 0/3`	Specifies an interface and enters interface configuration mode.
Step 17	**ethernet cfm mep level** *level-id* **[inward | outward domain** *domain-name***] mpid** *id* **vlan{any |** *vlan-id* **| ,** *vlan-id* **|** *vlan-id - vlan-id* **|** *vlan-id - vlan-id***}** **Example:** `Router(config-if)# ethernet cfm mep level 7 outward domain Customer mpid 701 vlan 100`	Provisions an interface as a domain boundary.

Step 18	**end**
	
	Example:
	`Router(config-if)# end`
	`Router#`
		Returns the router to the privileged EXEC mode.

Configuring and Enabling the Cross-Check Function (CE-A)

Perform this task to configure and enable cross-checking for a down MEP. This task requires you to configure and enable cross-checking on two devices. This task is optional.

SUMMARY STEPS

Step 1 **enable**

Step 2 **configure terminal**

Step 3 **ethernet cfm domain** *domain-name* **level** *level-id*

Step 4 **mep mpid** *mpid*

Step 5 **exit**

Step 6 **ethernet cfm mep crosscheck start-delay** *delay*

Step 7 **exit**

Step 8 **ethernet cfm mep crosscheck {enable | disable} domain** *domain-name*] **{port | vlan {***vlan-id* | *vlan-id - vlan-id* | *, vlan-id - vlan-id***}}**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router>enable`	Enables the privileged EXEC mode. Enter your password when prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters the global configuration mode.
Step 3	**ethernet cfm domain** *domain-name* **level** *level-id* **Example:** `Router(config)# ethernet cfm domain Customer level 7`	Defines a CFM maintenance domain at a specified level and enters Ethernet CFM configuration mode.
Step 4	**mep mpid** *mpid* **Example:** `Router(config-ecfm)# mep mpid 702`	Statically defines the MEPs within a maintenance association.
Step 5	**exit** **Example:** `Router(config-ecfm)# exit`	Returns the device to global configuration mode.
Step 6	**ethernet cfm mep crosscheck start-delay delay** **Example:** `Router(config)# ethernet cfm mep crosscheck start-delay 60`	Configures the maximum amount of time that the device waits for remote MEPs to come up before the cross-check operation is started.

	Command	Purpose
Step 7	**exit** **Example:** `Router(config)# exit`	Returns the device to privileged EXEC mode.
Step 8	**ethernet cfm mep crosscheck {enable | disable} domain** *domain-name*] **{port | vlan {***vlan-id* | *vlan-id - vlan-id* | **,** *vlan-id - vlan-id***}}** **Example:** `Router# ethernet cfm mep crosscheck enable domain cust4 vlan 100`	Enables cross-checking between the list of configured remote MEPs of a domain and MEPs learned through CCMs.

Configuring and Enabling the Cross-Check Function (CE-B)

SUMMARY STEPS

Step 1 **enable**

Step 2 **configure terminal**

Step 3 **ethernet cfm domain** *domain-name* **level** *level-id*

Step 4 **mep mpid** *mpid*

Step 5 **exit**

Step 6 **ethernet cfm mep crosscheck start-delay** *delay*

Step 7 **exit**

Step 8 **ethernet cfm mep crosscheck {enable | disable} domain** *domain-name*] **{port | vlan {***vlan-id* | *vlan-id - vlan-id* | , *vlan-id - vlan-id***}}**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router>enable`	Enables the privileged EXEC mode. Enter your password when prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters the global configuration mode.

	Command	Purpose
Step 3	**ethernet cfm domain** *domain-name* **level** *level-id* **Example:** `Router(config)# ethernet cfm domain Customer level 7`	Defines a CFM maintenance domain at a specified level and enters Ethernet CFM configuration mode.
Step 4	**mep mpid** *mpid* **Example:** `Router(config-ecfm)# mep mpid 702`	Statically defines the MEPs within a maintenance association.
Step 5	**exit** **Example:** `Router(config-ecfm)# exit`	Returns the device to global configuration mode.
Step 6	**ethernet cfm mep crosscheck start-delay delay** **Example:** `Router(config)# ethernet cfm mep crosscheck start-delay 60`	Configures the maximum amount of time that the device waits for remote MEPs to come up before the cross-check operation is started.
Step 7	**exit** **Example:** `Router(config)# exit`	Returns the device to privileged EXEC mode.
Step 8	**ethernet cfm mep crosscheck {enable | disable} domain** *domain-name*] **{port | vlan {***vlan-id* **|** *vlan-id - vlan-id* **| ,** *vlan-id - vlan-id***}}** **Example:** `Router# ethernet cfm mep crosscheck enable domain cust4 vlan 100`	Enables cross-checking between the list of configured remote MEPs of a domain and MEPs learned through CCMs.

Configuration Examples for Configuring Ethernet CFM for the Cisco ISR G2 Routers

The following two examples show configurations for a network. Configurations are shown not only for the Carrier Ethernet Cisco ISR G2 Routers, but also for the devices used at the access and core of the service provider's network.

- Example: Provisioning a Network, page 81
- Example: Provisioning Service, page 84

Example: Provisioning a Network

This configuration example shows only CFM-related commands. All commands that are required to set up the data path and configure the VLANs on the device are not shown. However, it should be noted that CFM traffic will not flow into or out of the device if the VLANs are not properly configured.

CE-A Configuration

```
!
ethernet cfm global
ethernet cfm ieee
!
ethernet cfm traceroute cache
ethernet cfm traceroute cache size 200
ethernet cfm traceroute cache hold-time 60
!
ethernet cfm mip auto-create level 7 vlan 1-4094
!
interface gigabitethernet3/2
 ethernet cfm mip level 7 vlan 101   <<<< Manual MIP
 ethernet cfm mep domain ServiceProvider-L4 mpid 401 vlan 101
 ethernet cfm mep domain OperatorA-L1 mpid 101 vlan 101
!
interface gigabitethernet4/2
 ethernet cfm mip level 1 vlan 101   <<<< Manual MIP
!
snmp-server enable traps ethernet cfm cc mep-up mep-down cross-connect loop config
snmp-server enable traps ethernet cfm crosscheck mep-missing mep-unknown service-up
```

U-PE A Configuration

```
!
ethernet cfm global
ethernet cfm ieee
!
ethernet cfm traceroute cache
ethernet cfm traceroute cache size 200
ethernet cfm traceroute cache hold-time 60
!
ethernet cfm mip auto-create level 7 vlan 1-4094
!
interface gigabitethernet3/2
 ethernet cfm mip level 7 vlan 101   <<<< Manual MIP
 ethernet cfm mep domain ServiceProvider-L4 mpid 401 vlan 101
 ethernet cfm mep domain OperatorA-L1 mpid 101 vlan 101
!
interface gigabitethernet4/2
 ethernet cfm mip level 1 vlan 101   <<<< Manual MIP
!
snmp-server enable traps ethernet cfm cc mep-up mep-down cross-connect loop config
snmp-server enable traps ethernet cfm crosscheck mep-missing mep-unknown service-up
```

PE-AGG A Configuration

```
ethernet cfm global
ethernet cfm ieee
ethernet cfm domain OperatorA-L1 level 1
mep archive-hold-time 65
  mip auto-create
  service MetroCustomer1OpA vlan 101
!
interface gigabitethernet3/1
 ethernet cfm mip level 1 vlan 101   <<<< Manual MIP
!
interface gigabitethernet4/1
 ethernet cfm mip level 1    <<<< Manual MIP
```

N-PE A Configuration

```
!
ethernet cfm global
ethernet cfm ieee
!
```

```
ethernet cfm traceroute cache
ethernet cfm traceroute cache size 200
ethernet cfm traceroute cache hold-time 60
!
ethernet cfm domain ServiceProvider-L4 level 4
 mep archive-hold-time 60
 mip auto-create
 service MetroCustomer1 vlan 101
  continuity-check
!
ethernet cfm domain OperatorA level 1
mep archive-hold-time 65
 mip auto-create
service MetroCustomer1OpA vlan 101
  continuity-check
!
interface gigabitethernet3/0
 ethernet cfm mip level 1  <<<< manual MIP
!
interface gigabitethernet4/0
 ethernet cfm mip level 4    <<<< manual MIP
!
snmp-server enable traps ethernet cfm cc mep-up mep-down cross-connect loop config
snmp-server enable traps ethernet cfm crosscheck mep-missing mep-unknown service-up
```

U-PE B Configuration

```
!
ethernet cfm global
ethernet cfm ieee
ethernet cfm traceroute cache
ethernet cfm traceroute cache size 200
ethernet cfm traceroute cache hold-time 60
!
ethernet cfm domain Customer-L7 level 7
 mip auto-create
 service Customer1 vlan 101 direction down
!
ethernet cfm domain ServiceProvider-L4 level 4
 mep archive-hold-time 60
 service MetroCustomer1 vlan 101
  continuity-check
!
ethernet cfm domain OperatorB level 2
 mip auto-create
 mep archive-hold-time 65
 service MetroCustomer1OpB vlan 101
  continuity-check
!
interface gigabitethernet1/0
 ethernet cfm mip level 7   <<<< manual MIP
!
interface gigabitethernet2/0
 ethernet cfm mip level 2   <<<< manual MIP
!
snmp-server enable traps ethernet cfm cc mep-up mep-down cross-connect loop config
snmp-server enable traps ethernet cfm crosscheck mep-missing mep-unknown service-up
```

PE-AGG B Configuration

```
ethernet cfm global
ethernet cfm ieee
!
ethernet cfm domain OperatorB level 2
```

```
 mep archive-hold-time 65
 mip auto-create
 service MetroCustomer1OpB vlan 101
!
interface gigabitethernet1/1
 ethernet cfm mip level 2   <<<< manual MIP
!
interface gigabitethernet2/1
 ethernet cfm mip level 2   <<<< manual MIP
```

N-PE B Configuration

```
!
ethernet cfm global
ethernet cfm ieee
!
ethernet cfm traceroute cache
ethernet cfm traceroute cache size 200
ethernet cfm traceroute cache hold-time 60
!
ethernet cfm domain ServiceProvider level 4
 mep archive-hold-time 60
 mip auto-create
 service MetroCustomer1 vlan 101
  continuity-check
!
ethernet cfm domain OperatorB level 2
 mep archive-hold-time 65
 mip auto-create
 service MetroCustomer1OpB vlan 101
  continuity-check
!
interface gigabitethernet1/2
ethernet cfm mip level 2    <<<< manual MIP
!
interface gigabitethernet2/2
 ethernet cfm mip level 4   <<<< manual MIP
!
snmp-server enable traps ethernet cfm cc mep-up mep-down cross-connect loop config
snmp-server enable traps ethernet cfm crosscheck mep-missing mep-unknown service-up
```

CE-B Configuration

```
!
ethernet cfm global
ethernet cfm ieee
ethernet cfm traceroute cache
ethernet cfm traceroute cache size 200
ethernet cfm traceroute cache hold-time 60
!
ethernet cfm domain Customer-L7 level 7
 service Customer1 vlan 101 direction down
  continuity-check
!
snmp-server enable traps ethernet cfm cc mep-up mep-down cross-connect loop config
snmp-server enable traps ethernet cfm crosscheck mep-missing mep-unknown service-up
```

Example: Provisioning Service

CE-A Configuration

```
!
ethernet cfm global
ethernet cfm ieee
```

```
ethernet cfm traceroute cache
ethernet cfm traceroute cache size 200
ethernet cfm traceroute cache hold-time 60
!
ethernet cfm domain Customer-L7 level 7
 service Customer1 vlan 101 direction down
  continuity-check
!
interface gigabitethernet3/2
 ethernet cfm mep domain Customer-L7 mpid 701 vlan 101
```

U-PE A Configuration

```
!
ethernet cfm global
ethernet cfm ieee
ethernet cfm traceroute cache
ethernet cfm traceroute cache size 200
ethernet cfm traceroute cache hold-time 60
!
ethernet cfm mip auto-create level 7 vlan 1-4094
!
ethernet cfm domain ServiceProvider-L4 level 4
 mep archive-hold-time 60
 service MetroCustomer1 vlan 101
  continuity-check
!
ethernet cfm domain OperatorA-L1 level 1
 mep archive-hold-time 65
 mip auto-create
 service MetroCustomer1OpA vlan 101
  continuity-check
!
interface gigabitethernet3/2
 ethernet cfm mip level 7 vlan 101   <<<< Manual MIP
 ethernet cfm mep domain ServiceProvider-L4 mpid 401 vlan 101
 ethernet cfm mep domain OperatorA-L1 mpid 101 vlan 101
!
interface gigabitethernet4/2
 ethernet cfm mip level 1 vlan 101   <<<< Manual MIP
```

PE-AGG A Configuration

```
ethernet cfm global
ethernet cfm ieee
ethernet cfm domain OperatorA-L1 level 1
mep archive-hold-time 65
  mip auto-create
  service MetroCustomer1OpA vlan 101
!
interface gigabitethernet3/1
 ethernet cfm mip level 1 vlan 101   <<<< Manual MIP
!
interface gigabitethernet4/1
 ethernet cfm mip level 1    <<<< Manual MIP
```

N-PE A Configuration

```
!
ethernet cfm global
ethernet cfm ieee
!
ethernet cfm traceroute cache
ethernet cfm traceroute cache size 200
ethernet cfm traceroute cache hold-time 60
```

```
!
ethernet cfm domain ServiceProvider-L4 level 4
 mep archive-hold-time 60
 mip auto-create
 service MetroCustomer1 vlan 101
  continuity-check
!
ethernet cfm domain OperatorA level 1
mep archive-hold-time 65
 mip auto-create
service MetroCustomer1OpA vlan 101
  continuity-check
!
interface gigabitethernet3/0
 ethernet cfm mip level 1    <<<< manual MIP
!
interface gigabitethernet4/0
 ethernet cfm mip level 4    <<<< manual MIP
 ethernet cfm mep domain OperatorA mpid 102 vlan 101
```

U-PE B Configuration

```
!
ethernet cfm global
ethernet cfm ieee
ethernet cfm traceroute cache
ethernet cfm traceroute cache size 200
ethernet cfm traceroute cache hold-time 60
!
ethernet cfm domain Customer-L7 level 7
 mip auto-create
 service Customer1 vlan 101 direction down
!
ethernet cfm domain ServiceProvider-L4 level 4
 mep archive-hold-time 60
 service MetroCustomer1 vlan 101
  continuity-check
!
ethernet cfm domain OperatorB level 2
 mep archive-hold-time 65
 service MetroCustomer1OpB vlan 101
  continuity-check
!
interface gigabitethernet1/0
 ethernet cfm mip level 7   <<<< manual MIP
 ethernet cfm mep domain ServiceProvider-L4 mpid 402 vlan 101
 ethernet cfm mep domain OperatorB mpid 201 vlan 101
!
interface gigabitethernet2/0
 ethernet cfm mip level 2   <<<< manual MIP
```

N-PE B Configuration

```
!
ethernet cfm global
ethernet cfm ieee
ethernet cfm traceroute cache
ethernet cfm traceroute cache size 200
ethernet cfm traceroute cache hold-time 60
!
ethernet cfm domain ServiceProvider level 4
 mep archive-hold-time 60
 mip auto-create
 service MetroCustomer1 vlan 101
  continuity-check
```

```
!
ethernet cfm domain OperatorB level 2
 mep archive-hold-time 65
 mip auto-create
 service MetroCustomer1OpB vlan 101
  continuity-check
!
interface gigabitethernet1/2
ethernet cfm mip level 2      <<<< manual MIP
!
interface gigabitethernet2/2
 ethernet cfm mip level 4     <<<< manual MIP
 ethernet cfm mep domain OperatorB mpid 202 vlan 101
```

CE-B Configuration

```
!
ethernet cfm global
ethernet cfm ieee
ethernet cfm traceroute cache
ethernet cfm traceroute cache size 200
ethernet cfm traceroute cache hold-time 60
!
ethernet cfm domain Customer-L7 level 7
 service Customer1 vlan 101 direction down
  continuity-check
!
interface gigabitethernet3/2
 ethernet cfm mep domain Customer-L7 mpid 702 vlan 101
```

Support for Y.1731 Performance Monitoring for EVC BD

Y.1731 Performance Monitoring (PM) provides a standard Ethernet PM function that includes measurement of Ethernet frame delay, frame delay variation, frame loss, and frame throughput measurements specified by the ITU-T Y-1731 standard and interpreted by the Metro Ethernet Forum (MEF) standards group.ITU-T Y.1731 feature supports key operation and maintenance standards that provide for automated end-to-end management and monitoring of Ethernet service by service providers.

Note This feature is supported only if you have purchased the DATA technology package functionality (*datak9*) licensing package. For more information about managing software activation licenses on the Cisco ISR and Cisco ISR G2 platforms, see http://www.cisco.com/en/US/docs/routers/access/sw_activation/SA_on_ISR.html.

Configuring a Sender MEP for a Single-Ended Ethernet Delay or Delay Variation Operation

Perform this task to configure a sender MEP on the source device.

Before You Begin

Time synchronization is required between the source and destination devices in order to provide accurate one-way delay (latency) or delay-variation measurements. Configure either Precision Time Protocol (PTP) or Network Time Pprotocol (NTP) on both the source and destination devices.

Note To display information about remote (target) MEPs on destination devices, use the **show ethernet cfm maintenance-points remote** command.

SUMMARY STEPS

Step 1 **enable**

Step 2 **configure terminal**

Step 3 **ip sla** *operation-number*

Step 4 **ethernet y1731 delay dmm domain** *domain-name* {**evc** *evc-id* | **vlan** *vlan-id*} {**mpid** *target-mp-id* | **mac-address** *target-address*} **cos** *cos* {**source** {**mpid** *source-mp-id* | **mac-address** *source-address*}}

Step 5 **clock sync**

Step 6 **aggregate interval** *seconds*

Step 7 **distribution {delay | delay-variation} one-way** *number-of-bins boundary*[,...,*boundary*]

Step 8 **frame interval** *milliseconds*

Step 9 **frame offset** *offset-value*

Step 10 **frame size** *bytes*

Step 11 **history interval** *intervals-stored*

Step 12 **max-delay** *milliseconds*

Step 13 **owner** *owner-id*

Step 14 **end**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables the privileged EXEC mode. Enter your password when prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters the global configuration mode.
Step 3	**ip sla** *operation-number* **Example:** `Router(config-term)# ip sla 10`	Begins configuring an IP SLAs operation and enters IP SLA configuration mode.

	Command	Purpose
Step 4	**ethernet y1731 delay dmm domain** *domain-name* {**evc** *evc-id* | **vlan** *vlan-id*} {**mpid** *target-mp-id* | **mac-address** *target-address*} **cos** *cos* {**source** {**mpid** *source-mp-id* | **mac-address** *source-address*}} **Example:** `Router(config-ip-sla)# ethernet y1731 delay dmm domain xxx evc yyy mpid 101 cos 4 source mpid 100`	Begins configuring a single-ended Ethernet delay operation and enters IP SLA Y.1731 delay configuration mode.
Step 5	**clock sync** **Example:** `Router(config-sla-y1731-delay)# clock sync`	(Optional) Indicates that the end points are synchronized and thus allows the operation to calculate one-way delay measurements.
Step 6	**aggregate interval** *seconds* **Example:** `Router(config-sla-y1731-delay)# aggregate interval 900`	(Optional) Configures the length of time during which the performance measurements are conducted and the results stored.
Step 7	**distribution {delay | delay-variation} one-way** *number-of-bins boundary[,...,boundary]* **Example:** `Router(config-sla-y1731-delay)# distribution delay-variation one-way 5 5000,10000,15000,20000,-1`	(Optional) Specifies measurement type and configures bins for statistics distributions kept.
Step 8	**frame interval** *milliseconds* **Example:** `Router(config-sla-y1731-delay)# frame interval 100`	(Optional) Sets the gap between successive frames.
Step 9	**frame offset** *offset-value* **Example:** `Router(config-sla-y1731-delay)# frame offset 1`	(Optional) Sets the value for calculating delay variation values.
Step 10	**frame size** *bytes* **Example:** `Router(config-sla-y1731-delay)# frame size 32`	(Optional) Configures padding size for frames.

	Command	Purpose
Step 11	**history interval** *intervals-stored* **Example:** `Router(config-sla-y1731-delay)# history interval 2`	(Optional) Sets the number of statistics distributions kept during the lifetime of an IP SLAs Ethernet operation.
Step 12	**max-delay** *milliseconds* **Example:** `Router(config-sla-y1731-delay)# max-delay 5000`	(Optional) Sets the amount of time an MEP waits for a frame.
Step 13	**owner** *owner-id* **Example:** `Router(config-sla-y1731-delay)# owner admin`	(Optional) Configures the owner of an IP SLAs operation.
Step 14	**end** **Example:** `Router(config-sla-y1731-delay)# end`	Exits to privileged EXEC mode.

What to Do Next

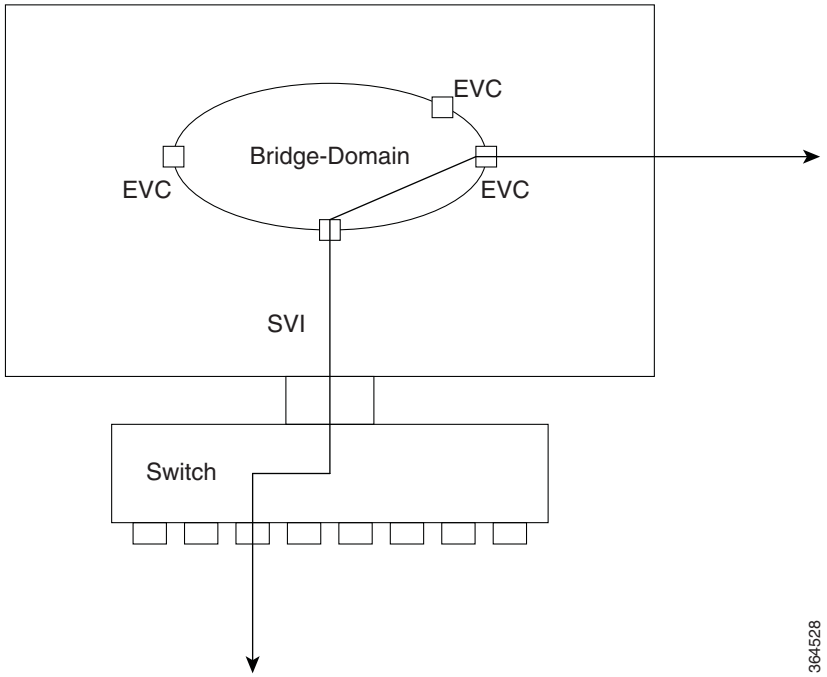
To add proactive threshold conditions and reactive triggering for generating traps, see the "Configuring Proactive Threshold Monitoring" module of the *IP SLAs Configuration Guide*.

When you are finished configuring proactive threshold monitoring for this operation, see the "Scheduling IP SLAs Operations" section to schedule the operation.

Support for Switch Virtual Interfaces (SVI) on ISR G2 Metro Ethernet BD

You can connect a SVI with a Metro Ethernet BD to re-direct the traffic from a switch port onto the BD and vice versa, as shown in Figure 1.

Figure 1 Re-directing the traffic from a SV1 onto the BD and vice versa



Once the SV1 is connected, packets coming into a switch port is re-directed to the SVI and onto the BD. On entering the BD, the source MAC address is learned and the packet is bridged. In the opposite direction, packets coming onto the BD from an EVC via the switch port are directed out the SVI.

Restrictions for SVI support on BDs

- Only one SVI may be associated with a BD.
- There is no EVC (i.e. service instance) configuration on an SVI.
- All packets on the BD, including those from EVCs, should be tagged, with the VLAN tag specifying the VLAN id of the SVI.
- Only access port configurations are supported.

Configuring SVI as Access Port

First you configure the switch port to add an access port SVI to a BD. After this you need to define the associated VLAN interface.

Note The BD id does not have to match the VLAN id in the dot1q tag, but all packets on the BD must be tagged with that VLAN number. So an EVC could be configured in which the BD id matches the VLAN id.

Configuration Examples to add an Access Port SVI to a BD

This example shows how to add an Access Port SVI to a BD:

```
interface GigabitEthernet4
 switchport access vlan 40
```

```
 no ip address
end
```

This example shows how to define the associated VLAN interface:

```
interface Vlan40
 no ip address
 bridge-domain 40
end
```

This example shows the BD id matching with the VLAN id:

```
interface GigabitEthernet8
 no ip address
 duplex auto
 speed auto
 service instance 40 ethernet
 encapsulation dot1q 40
 bridge-domain 40
 !
End
```

EVC Quality of Service (QoS)

For information about EVC QoS, see http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_mqc/configuration/xe-3s/qos-mqc-xe-3s-book/qos-evc.html.