JUNIPER NETWORKS

AMBASSADOR

DAY ONE: JUNOS® FUSION DATA CENTER UP AND RUNNING

Everything you need to know to deploy Junos Fusion Data Center: how to architect, manage, and operate it, and how it compares to other fabric technologies. Bridge the gap between theory and production networks today.

By Stefan Fouant

DAY ONE: JUNOS® FUSION DATA CENTER UP AND RUNNING

Junos Fusion Data Center represents a significant leap forward in the evolving technology of Ethernet fabric architectures that can manage large numbers of switches and Ethernet ports. *Day One: Junos Fusion Data Center Up and Running* gets you started with the technical underpinnings and components of Junos Fusion, and then follows you into the lab to get it up and running. After a few chapters you will have achieved connectivity between hosts, servers, racks, and infrastructure, with all the scalability and ease of management that today's modern data centers need to compete. It's day one and you have a data center to build.

"Modern data center architectures change about as often as the seasons. With virtualization and SDN at the center of it all, you need a scalable, agile solution. Stefan Fouant takes you on a journey through implementing the Junos Fusion Data Center architecture that can solve these modern-day challenges. His technically sound writing style and years of experience make it all seem easy, and he lays out a straight-forward configuration path that would make the Wizard of OZ jealous."

Scott Ware, Security Engineer, Juniper Ambassador, GSEC, JNCIS-SEC, JNCDA

IT'S DAY ONE AND YOU HAVE A JOB TO DO, SO LEARN HOW TO:

- Understand the basic principles of modern day data centers and why fabric architectures are increasingly important.
- Understand the concepts of underlay and overlay and how Junos Fusion Data Center fits into the equation.
- Understand how Junos Fusion compares to Virtual Chassis and Virtual Chassis Fabric, and when to deploy Junos Fusion Data Center.
- Understand the basic terminology and components of a Junos Fusion Data Center.
- Configure a Junos Fusion Data Center using either a single aggregation device or multiple aggregation devices, and learn how to add satellites into the topology.
- Manage the day-to-day operation of a Junos Fusion Data Center, including verification of satellite status, upgrading software on satellites, and removing satellites from the topology.
- Connect to hosts in a Junos Fusion Data Center and take advantage of advanced concepts such as VLAN Autosensing and Uplink Failure Detection.

ISBN 978-1-941441-63-3

Day One: Junos® Fusion Data Center Up and Running

by Stefan Fouant

Chapter 1: Introduction to Junos Fusion and the Modern Data Center 8

Chapter 2: Initial Junos Fusion Configuration with a Single Aggregation Device 22

Chapter 3: Scaling up with a Dual Aggregation Device Topology 41

Chapter 4: Junos Fusion Data Center Administration 59

Chapter 5: Connecting to Hosts in Junos Fusion Data Center 68

Conclusion: Data Center Simplicity 92

Appendix: Configurations 94

Resources 105

JUNIPER NETWORKS

\

Published by Juniper Networks Books
Author: Stefan Fouant
Technical Reviewers: Kevin Barker, Clay Haynes, Richard Hendricks, Greg Houde,
Editor in Chief: Patrick Ames
Copyeditor: Nancy Koerbel
Illustrator: Karen Joice

ISBN: 978-1-941441-63-3 (print)
Printed in the USA by Vervante Corporation.

ISBN: 978-1-941441-64-0 (ebook)

Version History: v1, December 2017
2 3 4 5 6 7 8 9 10

http://www.juniper.net/dayone

About the Author
Stefan Fouant is the Chief Architect at Copper River Information Technology with over 19 years of experience in the Service Provider and network security industries. He is a Juniper Ambassador and has worked with Juniper technologies since the inception of the M40 in 1998. He holds several patents in the area of DDoS detection and mitigation and is a co-author of drafts within the IETF DOTS working group relating to standardized signaling of coordinated DDoS attack filtering and mitigation mechanisms. He is a quadruple JNCIE (SP/ENT/SEC/DC), and a co-author of *Day One: Juniper Ambassadors Cookbook 2017*. Find Stefan on Twitter: @sfouant and also check out his blog at http://www.shortestpathfirst.net.

Author's Acknowledgments
I would like to give my sincerest thanks to Patrick Ames for the opportunity to be involved in writing this book, and for his steady hand in steering this project from beginning to end. I would also like to give a special thanks to my fellow Juniper Ambassadors, Clay Haynes and Kevin Barker, for their insightful technical review. I would also like to thank my manager Cybil Parker for graciously allowing me the time to work on this project, and to Lenny Giuliano and Jeff Kihn at Juniper for assisting me with acquiring the hardware that was required for the simulations. Last, but certainly not least, I would like to thank my girlfriend Michele for her patience during this project, without which this book would not be possible.

Welcome to Day One

This book is part of the *Day One* library, produced and published by Juniper Networks Books.

Day One books were conceived to help you get just the information that you need on day one. The series covers Junos OS and Juniper Networks networking essentials with straightforward explanations, step-by-step instructions, and practical examples that are easy to follow.

The *Day One* library also includes a slightly more comprehensive and longer suite of *This Week* books, whose concepts and test bed examples are more similar to a weeklong seminar.

You can obtain publications from either series in multiple formats:

- Download a free PDF edition at http://www.juniper.net/dayone.
- Get the ebook edition for iPhones and iPads from the iTunes/iBooks Store. Search for *Juniper Networks Books* or the title of this book.
- Get the ebook edition for any device that runs the Kindle app (Android, Kindle, iPad, PC, or Mac) by opening your device's Kindle app and going to the Kindle Store. Search for *Juniper Networks Books* or the title of this book.
- Purchase the paper edition at Vervante Corporation (www.vervante.com) for between $15-$30, depending on page length.
- Note that most mobile devices can also view PDF files.

What You Need to Know Before Reading This Book

Before reading this book, you should be familiar with the basic administrative functions of the Junos operating system, including the ability to work with operational commands and to read, understand, and change Junos configurations. There are several books in the *Day One* library devoted to learning Junos at http://www.juniper.net/dayone.

This book makes a few assumptions about you, the reader:

- You should be proficient in the use of Junos OS operation and configuration commands.
- You have a basic understanding of Layer 3 routing, as well as a good understanding of Internet Protocol version 4.
- You have a basic understanding of network design within a data center.
- You should be well versed in the concepts of Layer 2 switching as well as a how to configure VLANs for broadcast domain separation, including configuration of Integrated Routing and Bridging (IRB) interfaces for Inter-VLAN routing.
- You have a basic understanding of 802.3ad Link Aggregation as well as an understanding of the Link Aggregation Control Protocol (LACP).
- You have some background knowledge on Multi-Chassis Link Aggregation Groups (MC-LAG) and understand its operation when configured across multiple peers.
- In order to emulate the lab build out, you should have access to a minimum of two QFX10002-36Q or QFX10002-72Q switches for the aggregation devices, and a minimum of two satellites (see the *Models Supported* section in Chapter 1).

What You Will Learn by Reading This Book

- Understand the basic principles of modern day data centers and why fabric architectures are increasingly important.
- Understand the concepts of underlay and overlay and how Junos Fusion Data Center fits into the equation.
- Understand how Junos Fusion compares to Virtual Chassis and Virtual Chassis Fabric, and when to position Junos Fusion Data Center.
- Understand the basic terminology and components of a Junos Fusion Data Center.

- Configure a Junos Fusion Data Center using either a single aggregation device or multiple aggregation devices, and learn how to add satellites into the topology.
- Manage the day-to-day operation of a Junos Fusion Data Center, including verification of satellite status, upgrading software on satellites, and removing satellites from the topology.
- Connect to hosts in a Junos Fusion Data Center and take advantage of advanced concepts such as VLAN Autosensing and Uplink Failure Detection.

Resources

The last page of this *Day One* book contains links and resources from Juniper Networks and from the author to extend your knowledge and keep you up to date. Resource links are also provided within individual chapters as needed.

Chapter 1

Introduction to Junos Fusion and the Modern Data Center

Junos Fusion Data Center is Juniper Networks' latest innovation in the data center space and represents a significant technology leap in the continually evolving world of Ethernet fabric architectures. Fusion represents a simplistic yet extremely scalable methodology for managing large numbers of switches and Ethernet ports in a data center, and affords network designers and architects a seamless path forward once they've grown beyond the bounds of what Virtual Chassis or Virtual Chassis Fabric (VCF) architectures can provide. Furthermore, Fusion offers a highly-scalable alternative to the more complex IP Fabric configuration while yielding significant management advantages, especially if the idea of managing the entire switching infrastructure via a single-pane-of-glass is attractive.

This chapter investigates the requirements of the modern data center and examines why fabric architectures are becoming increasingly important. It looks at why software-defined concepts are taking hold within the data center, introduces the concept of underlays and overlays, and demonstrates how various fabric architectures fit into the picture. The chapter reviews the assorted Ethernet fabric architecture options that Juniper provides, and how they compare to the model provided in Junos Fusion Data Center, before finally moving on to a detailed description of the technical underpinnings and components that make up a Junos Fusion Data Center.

Subsequent chapters examine the configuration of a Junos Fusion Data Center, starting with a simple design consisting of a single aggregation device, before moving on to a more complex configuration involving multiple aggregation devices.

The Modern Data Center

It seems that everywhere you look, people are talking about data centers. Data centers – facilities where companies store, manage, and share data – are being built at a rapid rate, now more than at any time in recent history. The need for data has never been higher, and in order to meet that burgeoning need there has been explosive growth around data centers and the technologies that power them.

While businesses are continuing to drive higher demands for data and the underlying infrastructure to support it, there are continuing pressures to do so on minimal or decreasing IT budgets, causing both innovation and a tremendous amount of change over the past decade. Server virtualization has taken over every aspect of the data center in an effort to expedite the delivery of application infrastructure and make businesses more agile. Where it used to take weeks, if not months, to procure and configure a new server in this environment, it now takes only a matter of minutes to spin up a new virtualized instance. Storage has likewise been virtualized, and can be provisioned simultaneously alongside the virtual machines.

But the network has remained the long pole in the equation, impeding our overarching business objectives. It still takes a considerable amount of time to configure the network with the underlying connectivity that is required for the servers and applications, and this has been a reality for many IT operations personnel within the data center for ages…until recently, with the inception of simple Ethernet fabric architectures, software-defined networking (SDN) and the practical application of network virtualization concepts into modern day data center architectures.

By extending the concepts of SDN into data centers, you can achieve a network that is more agile and elastic, allowing you to pool resources in a fungible manner, eliminate costly silos, and allow the network to be automated. By automating the network and the connectivity it provides to hosts within the data center, the network is no longer a hindrance to business agility but rather a key enabler – instead of waiting days for the underlying network, the VLANs, the firewalls, and all the other network devices to be configured, you can automatically configure them as part of the provisioning process. When a virtual server and associated storage is spun up, you can simultaneously signal to the network that it should provision the underlying connectivity that is required by that server. Not only does it help us achieve our business goals more expeditiously, but a very attractive by-product is that it also eliminates expensive mistakes that are all but inevitable when you end up manually configuring network devices.

Underlays and Overlays

A basic premise underlying the effort to virtualize networks is separating the network into two planes – the underlay and the overlay. The underlay network is the physical network that provides basic connectivity, while the overlay is a virtual

network that is layered on top of the underlay. This concept of overlays isn't necessarily new – fundamentally, it's taking the concept of tunneling technologies such as generic routing encapsulation (GRE) and extending them into the data center such that you can interconnect all of your devices regardless of their location. Tunneling technologies such as GRE and Virtual Extensible LAN (VXLAN) allow you to transport both IP and non-IP traffic across a routed network, which allows for the entire network to behave as a virtualized switch within a data center or even across data centers.

Overlay networking has the potential to revolutionize the way data centers are architected because overlays can be automated and dynamically driven via SDN controllers or other automation tools, and therefore have the potential to eliminate the time-consuming task of manually configuring the network. You can now instantiate the network at a speed similar to that at which you are instantiating your compute and storage infrastructure, thus allowing businesses to focus on the business at hand, rather than the business of infrastructure.

However, in order to build an overlay solution in a data center, you must have a solid foundation to build upon, and this is where the underlay network becomes incredibly important. You must have a rock-solid underlay network that can scale, can adapt dynamically to network changes, and is simple to manage. You can think of it like building a house. The underlay is the foundation, and if it's not stable, everything else will come tumbling down.

This is where fabric architectures come into play.

Changing Traffic Patterns

Another fundamental shift that has happened within data centers is a change in traffic patterns, which have in turn necessitated the way that data center networks are architected.

Gone are the days of traditional client-server applications that simply generated traffic between an external user and a server in the data center in what is commonly referred to as a north-south traffic pattern. Instead, there's a dramatic shift of traffic patterns towards that of east-west traffic. This shifting traffic pattern is a result of applications that span across multiple distributed servers, for example, web servers that need to talk to mid-tier application servers, which in turn communicate with database servers. All of this has resulted in an explosion of east-west traffic, and has created considerable strain on traditional 3-tier architectures, which were more suited to north-south traffic patterns. These 3-tier architectures – consisting of a core layer, an aggregation later, and an access layer – were optimized for north-south traffic patterns, and are extremely inefficient when it comes to east-west traffic. When faced with east-west traffic patterns, these legacy architectures create not only wasted bandwidth, they have excessive latency through multiple hops, which impacts application performance.

Modern data centers are more commonly employing a fabric architecture that attempts to simplify and flatten the overall network architecture in a manner that is designed to efficiently handle the increase in east-west traffic, while at the same time maintaining the north-south connectivity required to the outside world.

Ethernet Fabric Architectures

Juniper has created a wide variety of fabric architectures across their switching portfolio to support data center networks of all sizes. Whether the need is to support a few dozen, or thousands, of ports, Juniper has a fabric solution.

Although there are many fabric architectures that Juniper supports, this chapter will strictly focus on Ethernet fabrics, as this is where Junos Fusion Data Center is positioned and this will allow us to make a direct comparison between them. Let's take a look at the three most commonly deployed Ethernet fabric architectures supported by Juniper – Virtual Chassis, Virtual Chassis Fabric, and now, Junos Fusion Data Center. But first let's get a better understanding of what an Ethernet fabric is all about.

Essentially, an Ethernet fabric is a collection of switches that are interconnected in a manner such that they can be characterized as being "woven into a fabric." Ethernet fabric architectures typically feature a switch, or switches, that act as the central management point for many other switches that are participating in the fabric. Ethernet fabrics are also typically depicted as having "any-to-any" connectivity and provide for deterministic latency across the fabric.

Ethernet fabrics may appear at first glance to be complex, but in fact they are actually quite simple, and key to their simplicity is that they allow you to flatten your network architecture. Gone are the days of cumbersome three-tier network designs consisting of the core layer, the distribution layer, and the access layer. In all Ethernet fabrics, you can reduce the number of tiers from three to two, and in some environments, you can go even further – collapsing the tiers down to just one. In addition, Ethernet fabrics do away with the complexity of having to configure loop prevention protocols such as Spanning Tree and eliminate the blocked ports that these protocols enforce. This allows all ports to be used at all times within your fabric architectures, yielding much higher bandwidth across the fabric than would otherwise be possible with these loop prevention protocols enabled.

Another attractive aspect of Ethernet fabrics is that all of the devices are centrally managed. So, if you have an Ethernet fabric comprised of a dozens or even hundreds of switches, you have single-pane-of-glass management capabilities dramatically streamlining and simplifying operations.

Virtual Chassis

Virtual Chassis was one of the first Ethernet fabric architectures created by Juniper with the introduction of the EX4200 switch in 2009. It was introduced as an alternative to some of the stacking technologies that were being sold by other vendors, and it allowed for up to 10 switches to operate as a single, logical device.

In order to allow all the devices to operate as a single device, one switch is designated as the Master routing engine, and another is designated as a Backup, much as we would see in a traditional chassis-based design consisting of multiple routing engines. Furthermore, the forwarding plane is extended across all members of the Virtual Chassis. In a sense, a Virtual Chassis behaves very much like the traditional chassis design, but with the flexibility of placing your FPCs outside the bounds of the physical chassis.

In order for switches to participate in a Virtual Chassis, devices are cabled in a ring configuration. Traffic received on one switch destined for another would travel around the ring until it reached the destination switch. Juniper uses a direct path algorithm to determine the best path and provide optimal traffic flow through the Virtual Chassis. A single control plane ensures that all the member devices share a common network topology.

Virtual Chassis, when configured within a data center, is optimized for relatively small data center environments.

MORE For more information on Virtual Chassis Technology, visit the Juniper TechLibrary: http://www.juniper.net/techpubs/en_US/junos14.1/topics/concept/virtual-chassis-ex4200-overview.html.

Virtual Chassis Fabric

Virtual Chassis Fabric (VCF) was an evolution of Virtual Chassis – taking the concepts of a single control plane across multiple switches but expanding upon it by allowing up to twenty switches to operate as a single, logical device. It was also the first Ethernet architecture offered by Juniper Networks to introduce a Clos topology, or what is more commonly referred to as a *spine and leaf topology*. A spine and leaf topology is essentially a hub and spoke design where the spines provide for connectivity amongst all of the leaves, and provide for end-to-end deterministic latency from any source to any destination.

Virtual Chassis Fabric is optimized for small to mid-sized data center environments.

MORE For more information on Virtual Chassis Fabric Technology, visit the Juniper TechLibrary: https://www.juniper.net/documentation/en_US/release-independent/junos/information-products/pathway-pages/qfx-series/product/index.html.

Junos Fusion Data Center

Junos Fusion Data Center is the latest Ethernet fabric technology introduced by Juniper, and it continues the evolution of previous Ethernet fabric architectures such as Virtual Chassis and Virtual Chassis Fabric. It provides benefits similar to all of the previous Ethernet fabric approaches in that it provides for minimal management and eliminates the need for Layer 2 loop prevention protocols, such as Spanning Tree Protocol, while at the same time allowing full bandwidth and all ports in the fabric to be utilized. Perhaps best of all, it provides these features while being incredibly simple to configure, yet it has scaling properties that can support some of the largest data center environments. At the time of this writing (Junos OS Release 17.3R1.10), a total of 64 switches acting as satellites are supported, providing for total port density of up to 3000 1GE/10GE server ports. But in the very near future, Junos Fusion will support up to 128 satellites for a total port density of up to 6000 1GE/10GE server ports. And, if an environment needs to scale beyond these limits, a point of delivery or PoD design could easily be incorporated by simply replicating a single Junos Fusion Data Center topology across multiple PoDs in a repeatable design pattern.

The Junos Fusion Data Center architecture is comprised of QFX10000 switches acting as *aggregation devices* and QFX5100 or EX4300 devices acting as *satellite devices*. Junos Fusion allows for significant expansion of the number of available interfaces on an aggregation device by allowing the ports on satellite devices to appear as extended ports on that aggregation device. These ports are then fully managed as if they were located directly on the aggregation device. In a sense, it's almost as if the satellite devices are remote linecards, or FPCs, from the perspective of the aggregation device. All satellite devices are therefore centrally managed via the aggregation device, resulting in simplicity of operations without ever having to resort to manually managing any of the individual satellites. In the Junos Fusion Data Center architecture, satellites are what typically provide access to the hosts and endpoints. Figure 1.1 illustrates satellite devices directly connected to the aggregation device in a hub and spoke topology.

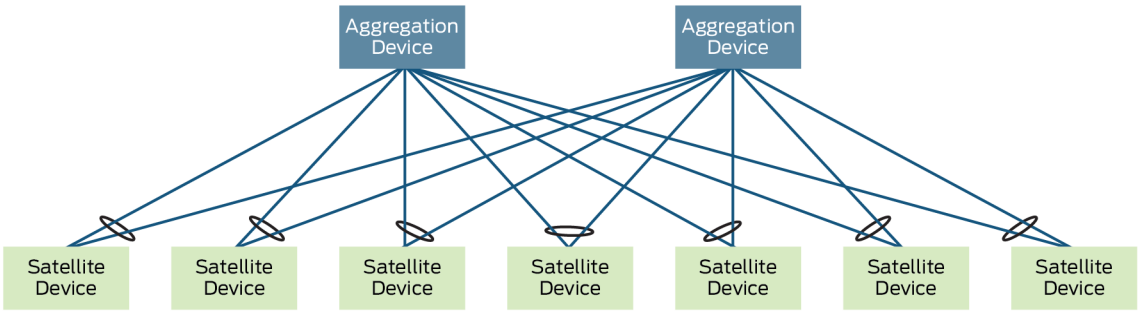

Figure 1.1 *Junos Fusion Data Center Architecture*

The astute reader may notice that in this instance Junos Fusion Data Center is just another variation of the modern spine and leaf model, as devices are interconnected in the typical 3-stage Clos fabric architecture. Multi-stage Clos fabric architectures provide for high resiliency, deterministic latency, high performance and scale, while at the same time minimize the complexity that is normally found in traditional, or legacy, 3-tier network designs consisting of an access layer, distribution layer, and core. You could therefore think of the aggregation devices as your spine layer, and the satellite devices as your leaf layer.

As was mentioned previously, it helps to think of the satellite devices simply as remote FPCs on an aggregation device, and one of the benefits of this is that they are managed via the aggregation device and therefore receive the full-feature set that is available on an aggregation device. This has incredibly important implications in modern data centers where switches at the leaf layer connecting to servers are typically based on merchant silicon chipsets. These switches tend to be fixed in their capabilities, and if a given feature set or function is not supported due to limitations of that chipset, these switches would need to be replaced with an alternative switch providing the required features. This is no longer the case with Junos Fusion Data Center. You can breathe new life into these leaf layer switching devices (in our case, the satellites), by extending the required feature set via the aggregation device. In Junos Fusion Data Center, the aggregation devices are QFX10000 switches, which are based on the Juniper Q5 chipset, which is *custom* silicon as opposed to merchant silicon found on the QFX5100s. This allows for a higher level of innovation and scale, and also allows newer services to be introduced more easily, in software rather than hardware. And, by extending these features to satellites, you greatly extend the life cycle of the equipment deployed within your data centers.

Although a Junos Fusion Data Center is most commonly deployed with multiple aggregation devices, it's perfectly acceptable to build a topology utilizing only a single aggregation device. The benefits of resiliency and redundancy are lost, but this might be perfectly acceptable in certain situations or design scenarios. When two aggregation devices are used, the satellites are multihomed to both aggregation devices, providing for greater load balancing as well as increased redundancy in the Junos Fusion Data Center topology.

The aggregation devices are responsible for all the management tasks for all the devices in the topology. Aggregation devices run traditional Junos OS software and provide the features and functionality of that software to all the elements within the topology, including the satellites and the network interfaces on the satellites.

Junos Fusion Control Plane

The control plane is maintained between the aggregation device and the satellites through a combination of several management protocols shown in Figure 1.2.

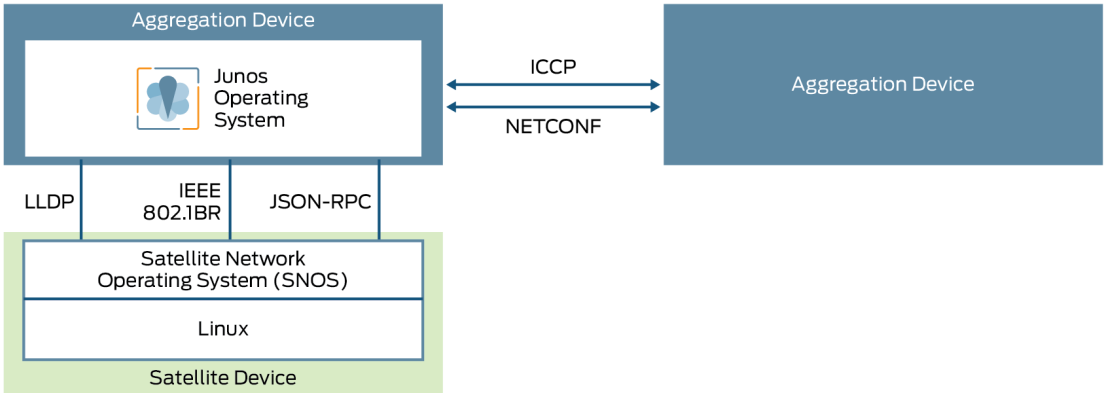

Figure 1.2 Junos Fusion Control Plane Protocols

The protocols in Figure 1.2 are:

- Link-Layer Discovery Protocol (LLDP) – provides for satellite device discovery and automatic provisioning. Additional information such as which port a satellite device is connected to and the firmware running on that satellite is collected via LLDP.

- IEEE 802.1BR+ – provides the forwarding plane between the aggregation devices and the satellites. The aggregation device assigns a 12-bit extended port (EPID) value to every extended port on satellites. Traffic is then switched throughout the Fusion architecture by simply appending the EPID value onto a packet of the intended destination port. In addition, the IEEE 802.1BR+ protocol handles management of all traffic-related aspects on satellites (configuration, statistics, port states, etc.).
- JSON-RPC – provides for management of all non-traffic related aspects on satellites (chassis, environment, upgrades).
- NETCONF – configuration synchronization between aggregation devices.
- Inter-Chassis Control Protocol (ICCP) – This is used to support the Multi-Chassis LAG (MC-LAG) supported between the two aggregation devices and ensure they are synchronized to support an active-active architecture. This gives the appearance that a satellite device is connected to a single, logical aggregation device, when in fact it may actually be connected to multiple aggregation devices.

To a large extent these protocols are automated through the Junos Fusion Data Center configuration, which hides much of the complexity from the administrator. This makes deploying a Junos Fusion Data Center incredibly simple, and data center environments using this technology can be up and running in a matter of minutes.

Ports in Junos Fusion Data Center

There are three different types of ports which are introduced in Junos Fusion Data Center: Cascade Ports, Uplinks, and Extended Ports. These ports and their relation between aggregation devices and satellites can be seen in Figure 1.3.

Cascade Ports

Cascade ports are physical interfaces on the aggregation device that interconnect with attached satellite devices and they process traffic sent and received between the aggregation device and satellite. All packets being transmitted between an aggregation device and a satellite device will traverse a cascade port. Cascade ports are typically10Gbps SFP+ ports or a 40Gbps QSFP+ ports, although other ports could be used in certain situations. In order to interconnect a satellite to an aggregation device, one of these ports must be configured as a cascade port.

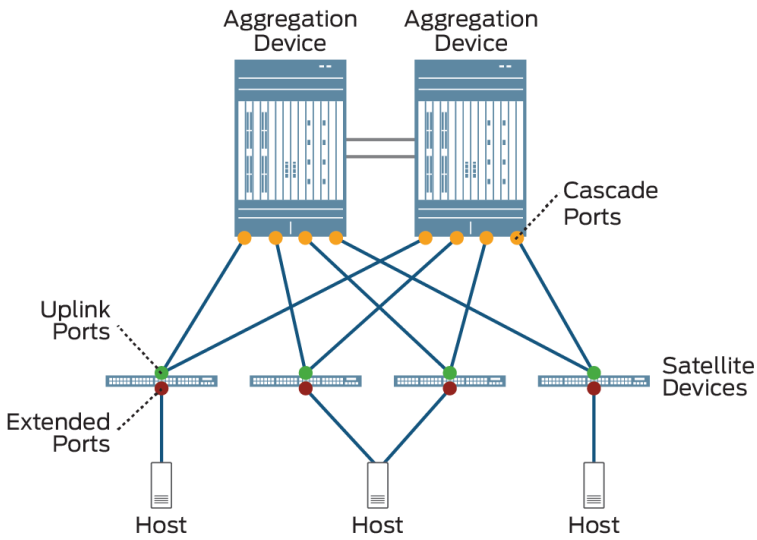

Figure 1.3 *Cascade, Uplink, and Extended Ports*

Uplink Ports

Uplink ports are physical interfaces on the satellite device that interconnect with attached aggregation devices. Typically, a satellite will have multiple uplink ports in the event of multiple connections to a single aggregation device, or more commonly, diverse connections across multiple aggregation devices. By utilizing multiple uplinks, a given satellite can achieve greater bandwidth and load-balancing and have higher levels of redundancy. All packets being transmitted between a satellite and an aggregation device traverse an uplink port. Uplink ports are typically a 10Gbps SFP+ port or a 40Gbps QSFP+ port, although other ports can be used.

Unlike cascade ports, which must be manually configured on an aggregation device, uplink ports are automatically created when a satellite is connected to a configured cascade port.

Extended Ports

Extended ports are physical interfaces on the satellite device that connect to hosts or other network segments. These are the primary, network-facing ports that transmit and receive all traffic going into a Junos Fusion Data Center.

These ports are so named because they appear as if they are directly connected to the aggregation device, therefore they are ports that are extended, by virtue of being controlled by the aggregation device.

In many cases, traffic received on an extended port is forwarded towards the aggregation device via the uplink port to the cascade port. Local switching can be configured on satellites, however, in order to switch traffic locally on a satellite should the source and destination MAC be reachable on that satellite and within the same broadcast domain.

Link aggregation groups can be configured on extended ports, and these can even span multiple satellite devices, allowing you to provide both link-level, as well as node-level, redundancy to connected hosts.

Traffic Switching Modes

There are two modes of operation within a Junos Fusion Data Center: *local switching* and *extended mode*.

The default operation is extended mode and is designed so that all features on the aggregation device can be extended to satellites. This is appealing because it allows you to extend any feature to the satellites, even features that might not be natively supported. This comes, however, at the expense of suboptimal traffic patterns, where a satellite device may have two hosts attached that want to communicate with each other, but have to take additional hops through the aggregation device in order to communicate.

Satellites can be configured for local switching mode if desired, where the network operator can optimize traffic forwarding between hosts connected to that satellite. This also comes with the trade-off that limited functionality may exist on that satellite, as traffic is no longer being forwarded to the aggregation device for traffic that is locally switched.

Fortunately, a Junos Fusion Data Center supports multiple modes, so some satellites can be configured for local switching while others run in extended mode, which gives the network operator a great deal of flexibility when deploying a Fusion architecture.

NOTE At the time of this writing, only Layer 2 local switching is supported on satellite devices. Layer 3 host forwarding will be supported in a future release, which will allow inter-VLAN routing and other routing operations to take place on satellite devices.

Understanding FPC Identifiers

When a satellite is added to a Junos Fusion Data Center, all of the ports on that satellite are added into the system as extended ports. However, in order to identify those ports, each satellite must be assigned a Flexible PIC Concentrator (FPC) slot identifier.

FPC identifiers for Junos Fusion Data Center are within the range of 65-254, and once assigned to a satellite extended ports on a given satellite are identified using the traditional fpc/pic/port format where the FPC variable is inserted based on the satellites assignment in the configuration. As an example, the first 10GE Ethernet extended port on a QFX5100-48S satellite that has been assigned to FPC slot 100 would be identified as xe-100/0/0.

Models Supported

At the time of this book's writing, the following device models are supported for each of the following Junos Fusion roles. (Always check for new developments.)

Aggregation Devices	Satellite Devices
QFX10002-72Q	EX4300-24T
QFX10002-36Q	EX4300-32F
	EX4300-48T
	QFX5100-24Q
	QFX5100-48S
	QFX5100-48T
	QFX5100-96S
	QFX5100-48SH
	QFX5100-48TH

NOTE The QFX5100-48SH and QFX5100-48TH are special SKUs designated by Juniper that come factory loaded with the satellite network operating system (SNOS) installed. These are ready to be inserted into a Junos Fusion Data Center environment without having to upgrade them to satellite software. They are eligible to have Junos installed on them through a license acquisition, as they are not running the Junos operating system by default.

Software Versions

On aggregation devices, the minimum software required to support Junos Fusion Data Center is Junos OS Release 17.2R1.

Once converted, satellite devices will run the satellite network operating system (typically on top of Windriver Yocto Linux or Linux Forwarding Engine OS (LFE OS) depending on the satellite model), but in order to become eligible for conversion into a satellite, they must be running at least Junos OS Release 14.1X53-D43 or higher. An exception to this are satellite models that come pre-shipped with the satellite software running on them, i.e. the QFX5100-48SH or the QFX5100-48TH – these will not require any upgrades unless the satellite software within the Junos Fusion Data Center is different than what is pre-installed.

MORE To see the latest software available for both aggregation devices as well as satellite devices, visit the Junos Fusion software download page: https://www.juniper.net/support/downloads/solutions/fusion/.

Licensing in Junos Fusion

Licensing in Junos Fusion is surprisingly simple. Essentially all you need are licenses on the aggregation devices for any features that are required within the Junos Fusion Data Center. Because the feature sets provided on the aggregation devices are inherited by the satellites, feature licenses on satellites are not required.

The one exception to this is if you have purchased the QFX5100-48SH or QFX5100-48TH variants with the satellite network operating system pre-installed. These devices are designed to be used as satellite devices, and as of Junos release 17.2R1 you will need to purchase and install a Junos Fusion license for them so they can be managed by the aggregation devices. Devices that are factory shipped with Junos OS can be converted to satellites without any additional licensing.

When installing these licenses for the QFX5100-48SH or QFX5100-48TH satellites, you will do so on the aggregation devices as these represent the single point of management for the entire Junos Fusion Data Center. In environments where dual aggregation devices are used, the license only needs to be installed on one as the license keys are fully synchronized between the two devices.

Single licenses can be used, or multi-pack licenses can be installed which can activate up to 128 satellite devices.

NOTE In the event that more satellites are provisioned than you have licenses installed for, the aggregation device will issue a warning indicating that there is a license limit violation.

Positioning Junos Fusion Data Center

When comparing fabric architectures, a question that often comes up is: Which architecture is best for a given environment? The answer depends on the size of the data center footprint, what level of redundancy is required, what feature sets are needed, and what types of ports will be required in the environment.

As a point of reference, let's compare VCF to Junos Fusion Data Center. They are both Ethernet fabric architectures providing similar capabilities in terms of single-pane-of-glass management and other benefits, but scaling and flexibility are the major differentiators. A key distinction is that Virtual Chassis Fabric is comprised primarily of QFX5100 or QFX5110 devices using the Broadcom Trident II chipset. Practically speaking, these are the only switches that are fully supported in a VCF because a proprietary Broadcom fabric protocol, called *HiGig*, is used for the

communication between VCF members. And although Juniper has extended VCF to operate in mixed mode, with other switches, such as the EX4300, operating in such a mode forces the VCF to operate in a reduced capacity. This dependency on proprietary protocols makes it more difficult for Juniper to use merchant silicon from other vendors in the architecture, and forces design constraints on the overall system.

On the other hand, Junos Fusion Data Center uses open standards that provide a better path for creating a fabric architecture that can be supported by virtually all chipset vendors. This gives Juniper much more flexibility in choosing merchant silicon from a wide variety of vendors, without being locked-in due to proprietary protocols.

Since a Junos Fusion Data Center architecture is comprised of both custom silicon (the Q5 ASIC on the QFX10000) and merchant silicon on the satellites, the open, standards-based IEEE 802.1BR bridge port extension protocol is used in order to allow for communication between the aggregation devices and the satellites.

Another key consideration is scale – a VCF can support up to 20 members. This means that at most you can support between 16 to 18 leaf switches in a VCF when accounting for the two to four spine devices. Junos Fusion Data Center, on the other hand, currently supports up to 64 satellite devices, and will support 128 satellites in the very near future. More importantly, the aggregation devices are not included as part of that count. Taking all these aspects into consideration, it's clear there is a dramatic difference when it comes to the size of the data center footprint that can be supported.

Furthermore, when it comes to satellite management, Junos Fusion has some very attractive features that introduce the concept of *software upgrade groups*, which allow you to perform rolling upgrades of satellites in your environment. This allows you to perform upgrades on just a portion of your network, rather than all at once, and increases availability within the entire data center. More about software upgrade groups is covered in Chapter 2.

Chapter 2

Initial Junos Fusion Configuration with a Single Aggregation Device

Now that you have some basic understanding of Junos Fusion Data Center, let's take a look at the initial configuration utilizing a single aggregation device. Although it's possible to build Junos Fusion with multiple aggregation devices right from the onset, and this is actually the preferred approach, it's important to have a good understanding of the initial configuration elements between an aggregation device and the satellites. Therefore, for purposes of thoroughly explaining the concepts, you will start with just a single aggregation device. In Chapter 3 you will expand your Junos Fusion Data Center by incorporating additional aggregation devices.

Preparing the Aggregation Device

Prior to configuring the aggregation device, you must ensure you are running a version of the Junos OS that is fully compatible with Junos Fusion Data Center. At a minimum, you need Junos OS Release 17.2R1 or higher.

The first thing you need to do to start the Junos Fusion configuration is to configure the cascade ports on the aggregation device, then add that port to a satellite upgrade group in order to manage the version of the satellite network operating system that will be installed on any given satellite. In the following example, we'll configure et-0/0/0 as our cascade port and assign the satellite on that cascade port to FPC 100. In addition, we'll assign the satellite to software upgrade group upgrade-group-1. (The utility of software upgrade groups is discussed later in this chapter.)

```
{master:0}[edit]
jedi@AD-1# set interfaces et-0/0/0 cascade-port

{master:0}[edit]
jedi@AD-1# set chassis satellite-management fpc 100 cascade-ports et-0/0/0

{master:0}[edit]
jedi@AD-1# set chassis satellite-management upgrade-groups upgrade-group-1 satellite 100

{master:0}[edit]
jedi@AD-1# show chassis
satellite-management {
    fpc 100 {
        cascade-ports et-0/0/0;
    }
    upgrade-groups {
        upgrade-group-1 {
            satellite 100;
        }
    }
}

{master:0}[edit]
jedi@AD-1# commit
2017-10-05 03:54:48 UTC: Running FIPS Self-tests
Veriexec is not enforced, FIPS mode not available
2017-10-05 03:54:48 UTC: FIPS Self-tests Skipped
configuration check succeeds
commit complete
```

Once our configuration is committed, you need to place the satellite software package onto the aggregation device so it can be downloaded onto the respective satellites once connected. As you can see from the following output, we have configured the satellite upgrade group, but there is no software associated with it:

```
{master:0}
jedi@AD-1> show chassis satellite upgrade-group
                                          Group               Device
Group         Sw-Version                  State          Slot State
__ungrouped__
upgrade-group-1                           in-sync
```

In order to associate a satellite software package with a given upgrade group, you need to place it on the aggregation device so it can subsequently be downloaded to our satellites.

NOTE It may be necessary to expand the size of the `/user` disk partition on the aggregation device in order to make room for the added software packages as the installed satellite software will be placed within this directory. This can be accomplished by using the `request system storage user-disk expand` command, which requires a reboot. In reality, what is actually happening with this command is that the `/var/home` directory is being expanded and symlinked to the `/user` directory for

storing satellite software. If you get any error message stating that there is not enough room to install the software package, this command is sure to help. Here's an example of issuing this statement on our lab's AD-1:

```
jedi@AD-1> request system storage user-disk expand
This command will change the size of /user and reboot the system
Do you want to continue? [yes,no] (no) yes

Proceeding with expand operation
Platform disk size changed to 4096M
Making link from /var/home to /user
Initiating reboot to complete the operation
Shutdown at Thu Oct 12 06:25:22 2017.

*** System shutdown message from root@AD-2 ***
```

Afterwards you can proceed to assign a software package to the software upgrade group by using the `request system software add` statement as shown in the following example. This is assuming you have already copied the satellite software onto the aggregation device, in this case to the /var/tmp directory:

```
{master:0}
jedi@AD-1> request system software add /var/tmp/satellite-3.1R1.4-signed.tgz upgrade-group upgrade-
group-1
Validating image /var/tmp/satellite-3.1R1.4-signed.tgz
Metatags extracted
warning: Version '3.1R1.4' already exist
Satellite package version is '3.1R1.4'
Using existing package
Provisioning group <upgrade-group-1> with satellite package version '3.1R1.4'
Request processed successfully
```

And let's see verification:

```
{master:0}
jedi@AD-1> show chassis satellite upgrade-group detail
Software upgrade group: upgrade-group-1
Software package version: 3.1R1.4

jedi@AD-1> show chassis satellite
                          Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    Standalone      et-0/0/0     present
```

You can see that upgrade-group-1 now has satellite software version 3.1R1.4 associated with it, and you can also notice that the satellite configured on et-0/0/0 is showing as `present`. Notice, however, that the device is still in a standalone state. This is because the software has not yet been downloaded to the satellite and therefore is not being fully managed by the aggregation device.

Before you can install the software on the satellite, you must prepare it first, which we'll do next.

Preparing the Satellite Device

In order for a satellite to be managed by an aggregation device, it must be running Satellite Network Operating System (SNOS) software. The following procedure outlines the proper preparation of a switch for the satellite software. Prior to this it is important to ensure that the switch has a version of Junos installed that will allow it to be converted into a satellite device. The initial Junos OS release that supports satellite conversion is Junos 14.1X53-D43 for both the QFX5100s as well as the EX4300s.

NOTE Although it is quite easy to push the satellite software down to a satellite via the aggregation device, it is possible to pre-install the satellite software manually on a switch. Furthermore, Juniper now has several SKUs in the QFX family, which can be ordered from the factory with satellite software pre-installed. The following step can be skipped for any devices that have satellite software pre-installed.

The first step in preparing a switch for conversion is ensuring that it is upgraded to at least Junos 14.1X53-D43 or higher – the minimum software version that allows for conversion. Assuming that has been taken care of, the next step is to zeroize the configuration. This process is important in order to ensure that the satellite has no configuration residing on it, and it can therefore receive instruction from an aggregation device that it should become a satellite and allow for the satellite software to be installed:

```
{master:0}
jedi@switch1> request system zeroize
warning: System will be rebooted and may not boot without configuration
Erase all data, including configuration and log files? [yes,no] (no) yes

warning: ipsec-key-management subsystem not running - not needed by configuration.
warning: zeroizing fpc0
```

Once the box has been zeroized it will reboot, and make itself available for instruction and software download from the aggregation device. So let's continue the process of installing the satellite software by switching over to the aggregation device.

Understanding Software Upgrade Groups

Satellite upgrade groups (SUGs) are a fundamental building block within Junos Fusion Data Center because they allow a simplistic centralized upgrade process of satellite devices. They allow you to designate which satellite software package should be associated with a given set of satellites; all satellites within a software upgrade group will run the same version of the satellite software.

Junos Fusion Data Center has been designed around the concept of *loose coupling*, meaning that satellites, although managed via the aggregation device, can each independently run different versions of the satellite software. A *tightly coupled* architecture would be one in which all the satellites needed to run the same version of the satellite software simultaneously, but there are many detractions to tightly coupled systems, one being downtime during upgrades. In comparison, Juniper's Virtual Chassis Fabric is an example of a tightly coupled architecture.

A Junos Fusion Data Center will often contain multiple software upgrade groups, and this is where you see the benefits of a loosely coupled system, allowing different satellites to simultaneously run different versions of the satellite software while allowing you to avoid downtime during the upgrade process. It enables you to perform rolling upgrades within your environment, maintaining connectivity to hosts during these upgrades. In environments where hosts are multi-homed to multiple satellites, through careful planning, you can upgrade one set of satellites while others are unaffected, as can be seen in Figure 2.1.

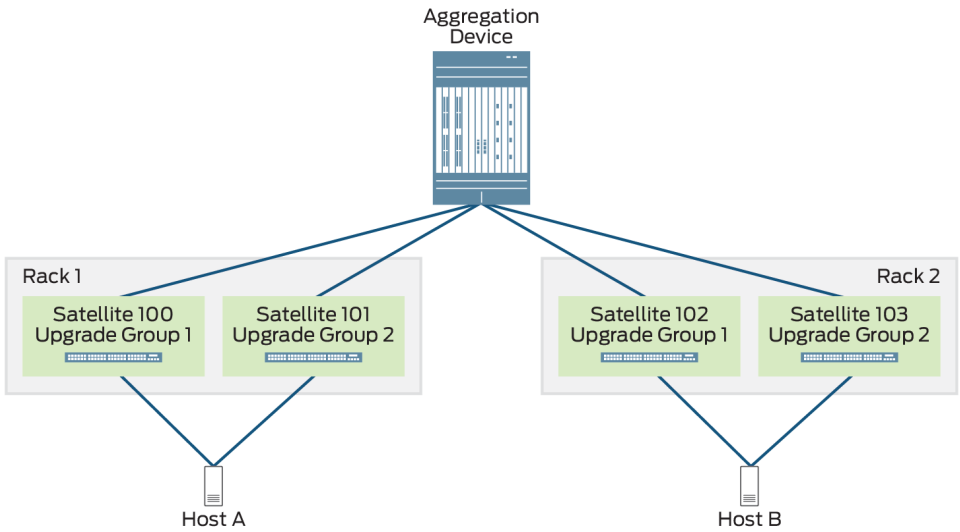

Figure 2.1 *Software Upgrade Groups*

In Figure 2.1 you can see that both Host A and Host B are multihomed across several satellites, which alternate their membership amongst software upgrade group 1 and software upgrade group 2. By performing upgrades of the satellite software only for satellites belonging to Software Upgrade Group 1, attached hosts can

maintain connectivity to the network through the alternate satellites in Software Upgrade Group 2. Furthermore, if the satellite software packages associated within a software upgrade group change, satellite members within that group are upgraded in a throttled manner. This minimizes the effect of major traffic disruptions from too many satellites being upgraded simultaneously.

Junos Fusion Data Center allows for as many software upgrade groups as the administrator desires. Software upgrade groups are required if using either *autoconversion* or *manual conversion* – the two most commonly employed approaches for installing satellite software.

Although software upgrade groups are not required for satellites that have had satellite software pre-installed, they do help simplify operations and provide for easy maintenance of satellites once these satellites are part of the Junos Fusion Data Center.

Installing Satellite Software onto the Satellite Device

There are three different methods by which satellite software can be installed onto a satellite device:

- (Recommended) Autoconversion – Satellite software is installed onto the satellite device automatically once the appropriate configuration is in place on the aggregation device and the satellite device has been cabled. This is the recommended mode for installing software onto satellites.
- Manual Conversion – Satellite software is installed onto the satellite device via a manual CLI command. This assumes that the appropriate configuration is in place on the aggregation device and the satellite has been cabled.
- Pre-installation – Satellite software is installed onto the satellite device prior to being cabled into the Junos Fusion Data Center. This can be pre-installed by Juniper at the factory, or can be installed by an administrator on a standalone switch by using the `request chassis device-mode satellite <URL-to-satellite software>` statement.

NOTE If the pre-installed method is used, the satellite device version is compared against the satellite device version of the software upgrade group that the satellite is associated with. If the satellite device is running a different version, the aggregation device will download the appropriate version of satellite software to the satellite based on the version specified in the software upgrade group, which will overwrite the pre-installed software package.

Manual Conversion

Let's first examine a manual installation of the satellite software onto a satellite to demonstrate what is taking place, and also look at the console output on a switch being converted so that you can get a better understanding of what is happening under the hood.

Assuming the proper versions of Junos are in place on your satellites and have been properly cabled to your configured aggregation device, you can manually install the software onto the satellite by using the following command on the aggregation device:

```
{master:0}
jedi@AD-1> request chassis satellite interface et-0/0/0 device-mode satellite
Mode change request initiated on interface et-0/0/0
Installing satellite-groups/upgrade-group-1/i386/host.tgz image on this device
```

Once you see the output indicate that it is installing the appropriate software on the satellite, you can turn to verification that the process is commencing:

```
{master:0}
jedi@AD-1> show chassis satellite
                          Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    ModeChanging    et-0/0/0     present
```

Here you can see that the _sd100 satellite in FPC slot 100, reachable via et-0/0/0, is in the Device State of `ModeChanging`, and the Port State is `present`. This is an indicator that the mode is changing from a standalone device to a satellite device.

Now let's take a look at what's happening from the perspective of the satellite via its console port to get a more thorough understanding of what's taking place. Special attention should be given to the sections highlighted in bold as these are key indicators that we are no longer in the traditional Junos OS. First you'll see that the system has received an interrupt and a request to shut down system processes and reboot. The first indicator that the satellite software is taking place is where the CLI output displays that a platform shutdown is taking place to downgrade the Host OS from Junos 14.1X53-D45.3 to Yocto Linux. Remember that the satellite network operating system generally runs on top of either Windriver Yocto Linux or Linux Forwarding Engine OS (LFE OS), depending on the satellite model:

```
Amnesiac (ttyd0)
login: Terminated

Oct  4 21:22:57 init: event-processing (PID 1138) exited with status=0 Normal Exit
Oct  4 21:22:57 init: l2cpd-service (PID 1767) exited with status=0 Normal Exit

Waiting (max 60 seconds) for system process `vnlru' to stop...done
Waiting (max 60 seconds) for system process `vnlru_mem' to stop...done
Waiting (max 60 seconds) for system process `bufdaemon' to stop...done
Waiting (max 60 seconds) for system process `syncer' to stop...
Syncing disks, vnodes remaining...0 0 0 0 done
```

```
syncing disks... All buffers synced.
Uptime: 16m0s
recorded reboot as normal shutdown
unloading fpga driver
Shutting down ACPI
Rebooting...
Stopping crond: [  OK  ]

Running guests on default URI: no running guests.
Stopping libvirtd daemon: [  OK  ]
Shutting down ntpd: [  OK  ]
Shutting down system logger: [  OK  ]
Shutting down sntpc: [  OK  ]
Stopping sshd: [  OK  ]
Stopping vehostd: [  OK  ]
Stopping watchdog: [  OK  ]
Stopping xinetd: [  OK  ]
Sending all processes the TERM signal... haveged: haveged: Stopping due to signal 15

[  OK  ]

Sending all processes the KILL signal... [  OK  ]

Plaform shutdown Downgrading Host OS from 14.1X53-D45.3 to Yocto
Converting /junos/images/shared/vjunos-user.img to QCOW2 v2...
Converting /junos/images/1/vjunos-config.img to QCOW2 v2...
Converting /junos/images/1/vjunos-data.img to QCOW2 v2...
Converting /junos/images/1/vjunos.img to QCOW2 v2...
Converting /junos/images/0/vjunos-config.img to QCOW2 v2...
Converting /junos/images/0/vjunos-data.img to QCOW2 v2...
Converting /junos/images/0/vjunos.img to QCOW2 v2...
[  OK  ]
Saving random seed:  [  OK  ]
Syncing hardware clock to system time [  OK  ]
Unmounting file systems:  [  OK  ]
init: Re-executiRE-FPGA-DRV: reboot notifier called with 0x0001
ng /sbin/init
RE-FPGA-DRV: Please standby while rebooting.
```

After the reboot are further indications that you are loading an entirely different operating system:

```
Filesystem type is ext2fs, partition type 0x83
i8042: No controller found
First Level Bootstrap using initramfs...
Mounting boot device UUID="1ff74b05-f429-4312-8a92-c0a97a839710"
Unpacking initrd.cpio.gz ....
850984 blocks
Unmount boot device UUID="1ff74b05-f429-4312-8a92-c0a97a839710"
Installation on QFX5_48X10G_6X40G
Install device is /dev/sda
/dev/sda size is 15272 Mb

------------------------------------------------------------------------
 Partition size requested  on /dev/sda
------------------------------------------------------------------------
Boot partition size   :   1024 MB
Rootfs partition size :   2048 MB
```

```
Var partition size    :   10152 MB
Swap partition size   :   597 MB
App partition size    :   1194 MB
------------------------------------------------------------------------
Deleting partition table on /dev/sda ...
2+0 records in
2+0 records out
1024 bytes (1.0 kB) copied, 0.00056306 s, 1.8 MB/s
Clear partition table.
Boot partition created
/dev/sda1 formatted for vfat with label=LINUX-BOOT.
Primary  partition created
mke2fs 1.42.8 (20-Jun-2013)
/dev/sda2 formatted for ext4 with label=APP_DISK.
Primary  partition created
mke2fs 1.42.8 (20-Jun-2013)
/dev/sda3 formatted for ext4 with label=ROOT.
Extended  partition created
Logical partition created
mke2fs 1.42.8 (20-Jun-2013)
/dev/sda5 formatted for ext4 with label=VAR.
Logical partition created
/dev/sda6 formatted for swap with label=SWAP.

------------------------------------------------------------------------
 Partitions created on /dev/sda
------------------------------------------------------------------------
Model: ATA TS16GHSD310 (scsi)
Disk /dev/sda: 16.0GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system     Flags
 1      8000kB  1280MB  1272MB  primary   fat16           boot
 2      1280MB  2474MB  1194MB  primary   ext4
 3      2474MB  4522MB  2048MB  primary   ext4
 4      4522MB  15.3GB  10.8GB  extended                  lba
 5      4522MB  14.7GB  10.2GB  logical   ext4
 6      14.7GB  15.3GB  598MB   logical   linux-swap(v1)

------------------------------------------------------------------------
Partitions created successfully.

Starting the platform installation..
Install the required boot components..
------------------------------------------------------------------------
Starting md5 validation:
Verifying ...
bzImage-intel-x86-64.bin: OK
initramfs.cpio.gz: OK
version.txt: OK
initrd.cpio.gz: OK
------------------------------------------------------------------------
Installed boot components
Installing rootfs files...
850984 blocks
Rootfs installation completed.
```

```
No software in /tmp/root/var/sw/applications requiring installation.
Updating fstabs and other config files ..
Platform Installation completed.

Changing boot device enable from  0x00 to 0x00
 0x3F 0x2f

INIT: version 2.88 booting

Mouting VAR to /var
Starting udev
cp: cannot stat '/etc/init.d/syslog.rsyslog': No such file or directory
 Adding system startup for /etc/init.d/rsyslog.
Creating platform specific configurations.
running pre rc install steps on QFX5_48X10G_6X40G
Installation finished. No error reported.
grub installed
Creating grub configuration file..
grub configuration file created.
Updating boot environment ..
boot environment updated
Completing the required package installation
Adding package group: pkg-qfx-5-hw
Adding package path: /var/sw/packages/pkg-qfx-5-hw

Adding package group: pkg-qfx-5-v44
Adding package path: /var/sw/packages/pkg-qfx-5-v44

Adding package group: applications
Adding package path: /var/sw/packages/applications

Installing package: fxbcm-util

Installing package: kernel-module-fxbcm

Installing package: platform-v44

Auditing: package
fxbcm-util
kernel-module-fxbcm
platform-v44
-------------------------------------------------
Running v44 application installer
-------------------------------------------------
Installing satellite package
Copying snos.tgz to /app_disk/software_update/snos.tgz
sw_installer
executing sw_installer
Compression successful: snos.tar
Invoking sw_app_installer to install the image
Starting the processing to install image snos.tar (option install-only)
Preparing directories for update
Extracted: libplatform_opus.so.1.0 in /usr/local/lib.new
Extracted: platform_xml_install.sh in /usr/local/bin.new
Extracted: libdrv_gfc0812.so.1 in /usr/local/lib.new
Extracted: libdrv_re_fpga.so.1.0 in /usr/local/lib.new
Extracted: libdrv_coretemp.so in /usr/local/lib.new
Extracted: logsync_check in /usr/local/bin.new
```

```
Extracted: app_logging.py in /usr/local/bin.new/common
Extracted: libdrv_opuspic.so.1 in /usr/local/lib.new
Extracted: core-file-mon.sh in /usr/sbin

< ----- Output truncated for brevity ----- >

Invoking platform_xml_install.sh to install XML files
BOARD_ID is 0bcd
BOARD_REV is 06
FAMILY is Opus
MODEL is CAYMUS
error: “hw.product.model” is an unknown key
Moving files to final location
Files moved
           update-rc.d log-sync.sh remove
update-rc.d: /etc/init.d/log-sync.sh exists during rc.d purge (continuing)
 Removing any system startup links for log-sync.sh ...
           update-rc.d log-sync.sh defaults 19 99
 Adding system startup for /etc/init.d/log-sync.sh.
  /etc/rc2.d/S19log-sync.sh -> ../init.d/log-sync.sh
  /etc/rc3.d/S19log-sync.sh -> ../init.d/log-sync.sh
  /etc/rc4.d/S19log-sync.sh -> ../init.d/log-sync.sh
  /etc/rc5.d/S19log-sync.sh -> ../init.d/log-sync.sh
  /etc/rc0.d/K99log-sync.sh -> ../init.d/log-sync.sh
  /etc/rc1.d/K99log-sync.sh -> ../init.d/log-sync.sh
  /etc/rc6.d/K99log-sync.sh -> ../init.d/log-sync.sh
/etc/app-version file is updated
Sync disk
Cleaning up old files
Cleaning up obseleted files
Obselected files deleted
Cleanup done
Sync disk
SW application updated successfully
Running SW version: 3.1R1.4
Moved: snos.tar.gz to /var/sd-sw/release/current
/var/sw/packages/applications
Application installed.
/
-----------------------------------------------
Installation for V44
Rebooting the system to complete the installation
-----------------------------------------------

INIT: Switchingstopping rsyslogd ... done
Stopping OpenBSD Secure Shell server: sshdno /usr/sbin/sshd found; none killed
Stopping domain name service: named.
Stopping system message bus: dbus.
Shutting down irqbalance: no irqbalance found; none killed
done
Stopping ntpd: no process in pidfile ‘/var/run/ntp.pid’ found; none killed
done
stopping rsyslogd ... done
Stopping internet superserver: xinetd.
Clearing ebtables rulesets: filter nat broute done. ok
Stopping crond: FAIL
Stopping S.M.A.R.T. daemon: smartd.
Deconfiguring network interfaces... ifdown: interface lo not configured
```

```
done.
Sending all processes the TERM signal...
Sending all processes the KILL signal...
Unmounting remote filesystems...
Deactivating swap...
Unmounting local filesystems...
Rebooting... RE-FPGA-DRV: reboot notifier called with 0x0001
RE-FPGA-DRV: Please standby while rebooting.


Booting `Juniper Linux'


Loading Linux ...

i8042: No controller found
First Level Bootstrap using initramfs...
Mounting boot device LABEL=LINUX-BOOT
Unpacking initrd.cpio.gz ....
850984 blocks
Unmount boot device LABEL=LINUX-BOOT

INIT: version 2.88 booting

Mouting VAR to /var
Starting udev
running pre rc steps on QFX5_48X10G_6X40G
Changing boot device enable from  0x00 to 0x00
 0x2F 0x2f
Executing application pre-initilization
Application Pre initialization.
Starting udev
Starting Bootlog daemon: bootlogd.

Populating dev cache
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.all.rp_filter = 1
kernel.core_uses_pid = 0
kernel.core_pattern = |/etc/init.d/zipcore.sh /var/tmp/corefiles/ %h.%e.%p.%t.core %e
fs.suid_dumpable = 2
vm.swappiness
INIT: Entering runlevel: 3

Starting system message bus: dbus.
starting rsyslogd ... done
Starting OpenBSD Secure Shell server: sshd
  generating ssh RSA key...
  generating ssh ECDSA key...
  generating ssh DSA key...
done.
Configuring network interfaces... done.
Starting domain name service: namedwrote key file "/etc/bind/rndc.key"
.
Starting irqbalance: done
Starting ntpd: done
starting rsyslogd ... done
Starting internet superserver: xinetd.
Starting crond: OK
```

```
Booting normal on QFX5_48X10G_6X40G
/root: 1.4 GiB (1492463616 bytes) trimmed
/var: 9.1 GiB (9749082112 bytes) trimmed
/app_disk: 1.1 GiB (1140158464 bytes) trimmed

Starting Watchdog ...
Executing application
Executing /usr/local/bin/cjob
Application initialization.
Stopping Bootlog daemon: bootlogd.

Wind River Linux 6.0.0.21 localhost console

localhost login:
Generated unique Monit id 924396c80355c87be3cb386dbc9ccda4 and stored to '/root/.monit.id'
Starting monit daemon with http interface at [localhost:2812]
```

At this point you can see that the switch, which was formerly a standalone switch, has gone through several reboots and in the process has undergone a conversion from the Junos OS to the satellite software package.

Now let's switch back to the aggregation device to see what has taken place. You can see from the next CLI output, after the satellite initially comes online, that it is in a Device State of `SyncWait`:

```
{master:0}
jedi@AD-1> show chassis satellite
                          Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    SyncWait        et-0/0/0     present    10/0
```

The satellite only stays in this Device State for a short time while it is waiting to have its control plane and forwarding state synchronized to the aggregation device. Once it is synchronized, the Device State changes to `Online`:

```
{master:0}
jedi@AD-1> show chassis satellite
                            Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    Online          et-0/0/0     online     10/2
```

Now let's verify that the ports on this satellite are appearing as ports on our aggregation device, taking the place of FPC 100 in our chassis:

```
{master:0}
jedi@AD-1> show interfaces terse | match -100
sd-100/0/0              up    up
sd-100/0/0.32770        up    up   eth-switch
ge-100/0/16             up    down
ge-100/0/18             up    down
ge-100/0/20             up    down
ge-100/0/22             up    down
xe-100/0/24             up    down
xe-100/0/26             up    down
xe-100/0/28             up    down
xe-100/0/30             up    down
```

```
et-100/0/48             up    up
et-100/0/49             up    up
```

Great. Now let's add yet another satellite into the mix, this time adding another Software Upgrade Group to the configuration. For the purposes of this example, let's stick with the manual conversion method.

First, let's look at the currently configured software upgrade groups and you can clearly see that only one has been configured thus far:

```
{master:0}[edit]
jedi@AD-1# run show chassis satellite upgrade-group
                              Group            Device
Group         Sw-Version                     State          Slot State
__ungrouped__
upgrade-group-13.1R1.4                       in-sync         100 version-in-sync
```

To configure our new satellite, first configure port et-0/0/1 as a cascade port, and then configure the second software upgrade group `upgrade-group-2`:

```
{master:0}[edit]
jedi@AD-1# set interfaces et-0/0/1 cascade-port

{master:0}[edit]
jedi@AD-1# edit chassis

{master:0}[edit chassis]
jedi@AD-1# set satellite-management fpc 101 cascade-ports et-0/0/1

{master:0}[edit chassis]
jedi@AD-1# set satellite-management upgrade-groups upgrade-group-2 satellite 101

{master:0}[edit chassis]
jedi@AD-1# show
satellite-management {
    fpc 100 {
        cascade-ports et-0/0/0;
    }
    fpc 101 {
        cascade-ports et-0/0/1;
    }
    upgrade-groups {
        upgrade-group-1 {
            satellite 100;
        }
        upgrade-group-2 {
            satellite 101;
        }
    }
}

{master:0}[edit chassis]
jedi@AD-1# commit and-quit
configuration check succeeds
commit complete
Exiting configuration mode
```

Once the configuration is committed, the next step is to load the satellite software into the aggregation device and associate it with upgrade-group-2, similar to what you did earlier with upgrade-group-1:

```
{master:0}
jedi@AD-1> request system software add /var/tmp/satellite-3.1R1.4-signed.tgz upgrade-group upgrade-
group-2
Validating image /var/tmp/satellite-3.1R1.4-signed.tgz
Metatags extracted
warning: Version '3.1R1.4' already exist
Satellite package version is '3.1R1.4'
Using existing package
Provisioning group <upgrade-group-2> with satellite package version '3.1R1.4'
Request processed successfully

{master:0}
jedi@AD-1> show chassis satellite upgrade-group
                                Group         Device
Group         Sw-Version                      State          Slot State
__ungrouped__
upgrade-group-13.1R1.4                        in-sync         100 version-in-sync
upgrade-group-23.1R1.4                        in-sync
```

You can see from the output that satellite software version 3.1R1.4 is now associated with upgrade-group-2, however, there are no slots that have been installed. This is because you still need to do the manual conversion. You can confirm this by looking at the satellite status:

```
{master:0}
jedi@AD-1> show chassis satellite
                            Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    Online          et-0/0/0     online     15/2
_sd101           101    Standalone      et-0/0/1     present
```

You can see that satellite _sd101 in slot 101 is still in a Device State of `Standalone`.

Time to request a manual conversion of the satellite software onto the satellite on et-0/0/1:

```
{master:0}
jedi@AD-1> request chassis satellite interface et-0/0/1 device-mode satellite
Mode change request initiated on interface et-0/0/1
Installing satellite-groups/upgrade-group-2/i386/host.tgz image on this device
```

Once the process has kicked off, you'll see that the Device State changes from `Standalone` to `ModeChanging` and the device goes down briefly as it reboots while it undergoes conversion. Finally, after the satellite software package has been installed and the device completes its reboot process, the satellite will return to normal functioning with a Device State of `Online` and a Port State of `Online`.

```
{master:0}
jedi@AD-1> show chassis satellite
                          Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    Online          et-0/0/0     online     15/2
_sd101           101    ModeChanging    et-0/0/1     present

{master:0}
jedi@AD-1> show chassis satellite
                          Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    Online          et-0/0/0     online     15/2
_sd101           101    ModeChanging    et-0/0/1     down

{master:0}
jedi@AD-1> show chassis satellite
                          Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    Online          et-0/0/0     online     15/2
_sd101           101    Online          et-0/0/1     online     18/2
```

Finally, let's verify that the ports on this satellite are now appearing as ports on the aggregation device, taking the place of FPC 101 in the chassis:

```
{master:0}
jedi@AD-1> show interfaces terse | match 101/
ge-101/0/0              up    down
sd-101/0/0              up    up
sd-101/0/0.32770        up    up   eth-switch
ge-101/0/2              up    down
ge-101/0/4              up    down
ge-101/0/6              up    down
ge-101/0/18             up    down
ge-101/0/19             up    down
ge-101/0/20             up    down
ge-101/0/21             up    down
ge-101/0/40             up    down
ge-101/0/41             up    down
ge-101/0/42             up    down
ge-101/0/43             up    down
ge-101/0/44             up    down
ge-101/0/45             up    down
ge-101/0/46             up    down
ge-101/0/47             up    down
et-101/0/48             up    up
et-101/0/49             up    up
```

Autoconversion

In this next example let's add a third satellite into our Junos Fusion Data Center. In this case, we'll be adding a QFX5100-24Q connected to et-0/0/2 and adding it to the first upgrade group created, namely upgrade-group-1. For the most part, the configuration is identical to the previous example except we'll be adding autoconversion to the new satellite. With autoconversion, you no longer have to manually

request that satellite software be installed onto the satellite – it takes place immediately upon committing the configuration, assuming the device is properly cabled to the aggregation device and the Junos Fusion Data Center is properly configured:

```
{master:0}[edit]
jedi@AD-1# set interfaces et-0/0/2 cascade-port

{master:0}[edit]
jedi@AD-1# edit chassis

{master:0}[edit chassis]
jedi@AD-1# set satellite-management fpc 102 cascade-ports et-0/0/2

{master:0}[edit chassis]
jedi@AD-1# set satellite-management upgrade-groups upgrade-group-1 satellite 102
```

And configure satellite 102 to be automatically converted:

```
{master:0}[edit chassis]
jedi@AD-1# set satellite-management auto-satellite-conversion satellite 102
```

Let's take a look at the full configuration and then commit it:

```
{master:0}[edit chassis]
jedi@AD-1# show
satellite-management {
    fpc 100 {
        cascade-ports et-0/0/0;
    }
    fpc 101 {
        cascade-ports et-0/0/1;
    }
    fpc 102 {
        cascade-ports et-0/0/2;
    }
    upgrade-groups {
        upgrade-group-1 {
            satellite [ 100 102 ];
        }
        upgrade-group-2 {
            satellite 101;
        }
    }
    auto-satellite-conversion {
        satellite 102;
    }
}

{master:0}[edit chassis]
jedi@AD-1# commit and-quit
configuration check succeeds
commit complete
Exiting configuration mode
```

At this point, you should quickly see satellite 102 transitioning from a `Standalone` device state to a device that is part of the Junos Fusion Data Center:

```
{master:0}
jedi@AD-1> show chassis satellite
                            Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    Online          et-0/0/0     online     15/2
 sd101           101    Online          et-0/0/1     online     18/2
_sd102           102    ModeChanging    et-0/0/2     down
```

The device reboots and is converted to the satellite software package, and then enters into a Device State of `SyncWait`:

```
{master:0}
jedi@AD-1> show chassis satellite
                            Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    Online          et-0/0/0     online     15/2
_sd101           101    Online          et-0/0/1     online     18/2
_sd102           102    SyncWait        et-0/0/2     present    2/0
```

And after a few minutes, the device is fully synchronized:

```
{master:0}
jedi@AD-1> show chassis satellite
                            Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    Online          et-0/0/0     online     15/2
_sd101           101    Online          et-0/0/1     online     18/2
_sd102           102    Online          et-0/0/2     online     2/2
```

Let's confirm that the ports associated with the newly added satellite are appearing as additional FPC ports on the aggregation device:

```
{master:0}
jedi@AD-1> show interfaces terse | match /102/
et-102/0/0              up    up
sd-102/0/0              up    up
sd-102/0/0.32770        up    up   eth-switch
et-102/0/1              up    up
```

Lastly, confirm that the multiple software upgrade groups are working as expected:

```
{master:0}
jedi@AD-1> show chassis satellite upgrade-group detail
Software upgrade group: upgrade-group-1
Software package version: 3.1R1.4
  Slot    Device State
  100     version-in-sync
  102     version-in-sync
```

Pre-Installation

In some cases, it's preferable to simply pre-install the satellite network operating system onto a switch prior to connecting it into a Junos Fusion Data Center. This will often be unnecessary, but might be the case if a device is going to be staged in one location before being shipped for RMA purposes, or you might run into issues when pushing down the software from the aggregation device. The following command can be used to pre-install the software and get the satellite functioning within the Junos Fusion Data Center. The example below takes place from the CLI of an EX4300 satellite, using the `request chassis device-mode satellite URL-to-satellite-software` command:

```
root# request chassis device-mode satellite satellite-ppc-3.2R1.1-signed.tgz
Fetching package...
Extracting image /root/satellite-ppc-3.2R1.1-signed.tgz
Metatags extracted
Satellite package version is '3.2R1.1'
Using host.tgz for ppc from the package
all conditions passed
Proceeding with conversion to satellite device
```

It is important to make sure you have the appropriate software if you are planning to pre-install the satellite network operating system on a device. In this example, a PPC version of software is being placed on the satellite, which is a slightly different package than the one that would be installed if it was being pushed down from the aggregation device.

NOTE Always make sure to check the Junos Platforms Software Downloads page to check for the latest or most appropriate version: https://www.juniper.net/support/downloads/group/?f=junos.

Chapter 3

Scaling up with a Dual Aggregation Device Topology

This chapter adds an additional aggregation device to our initial configuration of Junos Fusion Data Center, allowing a higher level of redundancy and resiliency, but also enabling a better load balancing of traffic from satellite to satellite by taking advantage of diverse paths through multiple aggregation devices. In the current version of Junos Fusion Data Center (at the time of this book's writing), load balancing through multiple aggregation devices is achieved by running Multi-Chassis Link Aggregation Group (MC-LAG) on the aggregation devices. The configuration is seamless and very easy to configure, and also fully transparent to satellites. A given satellite has no concept or understanding that it is in fact connecting to more than one aggregation device – from its perspective, it is simply a LAG connection.

A few new satellites will be added to the infrastructure you built in Chapter 2, in order to demonstrate the difference in configuration when dealing with a *dual aggregation device* topology. Once complete, the topology should look like the one illustrated in Figure 3.1.

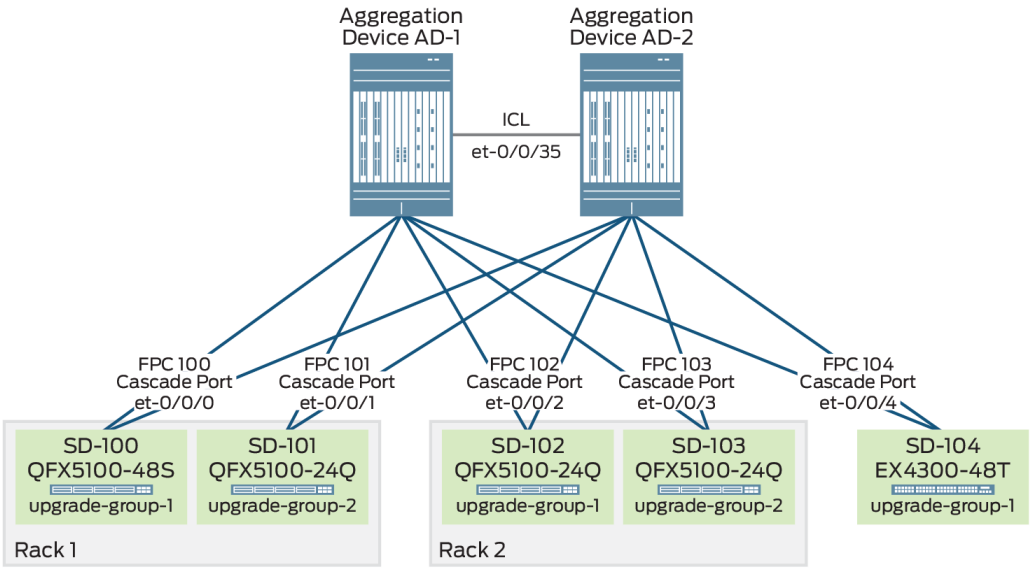

Figure 3.1 *Dual Aggregation Device Topology*

You can increase the bandwidth and load-balancing characteristics from satellite to satellite in much the same way as you can with any spine and leaf fabric architecture, by increasing the number of links between a satellite and an aggregation device. Furthermore, you can increase the width of the spine by increasing the number of aggregation devices, which also increases resiliency.

NOTE Not only is it entirely possible to build the Junos Fusion Data Center with multiple aggregation devices right from the onset, that is the preferred approach! But in this book the fabric architecture is built with a single aggregation device in order to better demonstrate the concepts.

Let's take a look at the configuration elements required to turn our single aggregation device Junos Fabric Data Center topology into a dual aggregation device redundant topology.

Dual Aggregation Device Architecture

In order to support two aggregation devices, and allow for satellites to be multihomed across multiple aggregation devices, commit synchronization must be enabled and certain elements of the configuration must be mirrored. Furthermore, a redundancy group must be configured which enables the two devices to participate in an MC-LAG using the Inter-Chassis Control Protocol (ICCP). This enables satellites to be multihomed across the two aggregation devices and ensures that these multiple links are treated as one within the Junos Fusion Data Center.

An interchassis Link (ICL) must be configured between the two aggregation devices, which allows for ICCP protocol messages to be exchanged and for data to be forwarded when traffic arrives on one aggregation device with a destination that is reachable via the other. Junos Fusion Data Center automates the ICCP configuration, removing much of the complexity of configuring and simplifying the deployment.

Configuring Commit Synchronization

Assuming both of the aggregation devices are online and reachable, the first thing to do is to configure commit synchronization between these two peers. Commit synchronization is used in the Junos Fusion Data Center solution to simplify the administration of multiple aggregation devices. Oftentimes there are configuration elements that need to match across all of the aggregation devices. Configuration synchronization simplifies the administration of the entire Junos Fusion Data Center by allowing an administrator to enter commands in a configuration group on one of the aggregation devices and then through `apply-groups` have this configuration replicated to other aggregation devices that are part of the Junos Fusion. This eliminates the time-consuming task of manually replicating these elements across all the aggregation devices. Configuration groups are used extensively throughout the Junos Fusion Data Center architecture in order to streamline and simplify the overall management of the solution.

Let's begin this synchronization by using the following configuration statements, with a specified user account to connect from one peer to the other. In this example, the IP address being used is the management address assigned to the em0 interface on the routing engine of the peering aggregation device:

```
{master:0}[edit]
jedi@AD-1# set system commit peers-synchronize

{master:0}[edit]
jedi@AD-1# set system commit peers 172.16.110.25 user sync_user

{master:0}[edit]
jedi@AD-1# set system commit peers 172.16.110.25 authentication "$9$jwkqf0OReK8hSs4aJDjCtpBhSKM8-bs"

{master:0}[edit]
jedi@AD-1# commit
warning: No valid groups are applied to export sync_peers.conf
2017-10-19 09:59:31 UTC: Running FIPS Self-tests
Veriexec is not enforced, FIPS mode not available
2017-10-19 09:59:31 UTC: FIPS Self-tests Skipped
configuration check succeeds
commit complete
```

NOTE Although we could have simply used the root user for the configuration above, the author strongly believes that it is best practice to configure a secondary user for this purpose. This will make automatic configuration changes pushed

through this synchronization process easier to audit and decipher through log analysis. This user should exist on both aggregation devices. The configuration for this sync_user is displayed below:

```
jedi@AD-1# show system login
user sync_user {
    uid 2001;
    class super-user;
    authentication {
        encrypted-password "$6$0rUG7iZT$NQGZp9G9XjiXN4V7ef1ils4kYs7gBQ4KV8myMx7DU1BAoyr/
BdOwfdgOKlXPgvQHK7eqQCy.j2XeF4ne5lSBm/"; ## SECRET-DATA
    }
}
```

You need to perform a similar configuration on AD-2, to ensure that any configuration that is committed on AD-2 will likewise be synchronized with AD-1:

```
{master:0}[edit]
jedi@AD-2# set system commit peers-synchronize

{master:0}[edit]
jedi@AD-2# set system commit peers 172.16.110.24 user sync_user

{master:0}[edit]
jedi@AD-2# set system commit peers 172.16.110.24 authentication "$9$ZMDjqAtOSyK1Rds2gJZn/9p1RylKXNd"

{master:0}[edit]
jedi@AD-2# commit
warning: No valid groups are applied to export sync_peers.conf
2017-10-19 02:40:31 UTC: Running FIPS Self-tests
Veriexec is not enforced, FIPS mode not available
2017-10-19 02:40:31 UTC: FIPS Self-tests Skipped
configuration check succeeds
commit complete
```

Redundancy Group Configuration

At the time of this writing, Junos Fusion Data Center supports a dual aggregation device topology. In order to achieve this, an MC-LAG is defined which uses ICCP to allow for communication and forwarding of traffic between the aggregation devices. In a Junos Fusion Data Center architecture, the only peers participating in the MC-LAG are the aggregation devices. Satellites connected to the aggregation devices are not aware that they are connecting to hosts providing an MC-LAG, in fact, to satellites, these simply appear as standard LAG connections.

The basic building block to enabling an MC-LAG in a Junos Fusion Data Center is to configure a redundancy group. Although this is not the traditional way that an MC-LAG is configured, the entire process has been streamlined so that ICCP is automatically provisioned through the redundancy group configuration. This is a welcome enhancement for any of you who have ever configured MC-LAG in the past as you are likely aware of the myriad configuration elements and difficulty traditionally required to get an MC-LAG up and running between peers.

MORE Although a detailed explanation of MC-LAG and ICCP is beyond the scope of this book, the interested reader is encouraged to look at the book, "*Juniper MX Series: A Comprehensive Guide to Trio Technologies on the MX 2nd Edition*," written by Douglas Hanks, Harry Reynolds, and David Roy (O'Reilly Media, 2016). See: https://www.juniper.net/us/en/training/jnbooks/oreilly-juniper-library/mx-series/. It's a great read and highly recommended. (I would know – I happened to be a technical reviewer!)

Let's start by configuring a redundancy group, which includes our two aggregation devices as member devices. Both members will share the same redundancy group number, but each aggregation device will have a unique Chassis-ID. Both aggregation devices need to have an ICCP link connected between the two of them, in this case et-0/0/35, but this can be any arbitrary interface or LAG within the system. It is considered a best practice to have multiple ICCP links between MC-LAG peers for redundancy and for increased bandwidth. Aggregated Ethernet interfaces are typically used for this, but for the purposes of this book let's keep things simple.

Start with the first aggregation device:

```
{master:0}[edit chassis satellite-management]
jedi@AD-1# set redundancy-groups chassis-id 1

{master:0}[edit chassis satellite-management]
jedi@AD-1# set redundancy-groups dc-pod-1 redundancy-group-id 1

{master:0}[edit chassis satellite-management]
jedi@AD-1# set redundancy-groups dc-pod-1 peer-chassis-id 2 inter-chassis-link et-0/0/35

{master:0}[edit chassis satellite-management]
jedi@AD-1# set redundancy-groups dc-pod-1 satellite all
```

The above configuration would be enough to get ICCP running between the two aggregation devices, however for more rapid failure detection between them you want to configure Bidirectional Forwarding Detection (BFD). The settings below are adequate for most environments but should be fine-tuned to meet your individual requirements:

```
{master:0}[edit chassis satellite-management]
jedi@AD-1# set redundancy-groups dc-pod-1 peer-chassis-id 2 liveness-detection minimum-interval 2000

{master:0}[edit chassis satellite-management]
jedi@AD-1# set redundancy-groups dc-pod-1 peer-chassis-id 2 liveness-detection multiplier 3

{master:0}[edit chassis satellite-management]
jedi@AD-1# set redundancy-groups dc-pod-1 peer-chassis-id 2 liveness-detection transmit-
interval minimum-interval 2000

{master:0}[edit chassis satellite-management]
jedi@AD-1# commit
warning: No valid groups are applied to export sync_peers.conf
configuration check succeeds
commit complete
```

The last thing to do before committing the configuration is to configure the et-0/0/35 interface used for the inter-chassis-link with the ability to trunk all VLANs. If you leave this out, the commit will error out with a warning that the 'bridge' or 'ethernet-switching' family on unit 0 must be present for ICL trunk interface to function properly if `auto-vlan-provisioning` is enabled. The default behavior of Junos Fusion Data Center configured with a redundancy group is to utilize `auto-vlan-provisioning` between the two aggregation devices, thus this step is necessary to bring up the ICL:

```
{master:0}[edit interfaces et-0/0/35]
jedi@AD-1# set unit 0 family ethernet-switching interface-mode trunk

{master:0}[edit interfaces et-0/0/35]
jedi@AD-1# set unit 0 family ethernet-switching vlan members all
```

Finally, let's commit the configuration:

```
{master:0}[edit chassis satellite-management]
jedi@AD-1# commit
warning: No valid groups are applied to export sync_peers.conf
configuration check succeeds
commit complete
```

Note the warning here stating that no valid groups are being exported to our sync_peers. This is because we haven't configured the groups, which gets done in the next section. For now, let's take a look at our final redundancy group configuration:

AD-1

```
chassis {
    satellite-management {
        redundancy-groups {
            chassis-id 1;
            dc-pod-1 {
                redundancy-group-id 1;
                peer-chassis-id 2 {
                    inter-chassis-link et-0/0/35;
                    liveness-detection {
                        minimum-interval 2000;
                        multiplier 3;
                        transmit-interval {
                            minimum-interval 2000;
                        }
                    }
                }
                satellite all;
            }
        }
    }
}
interfaces {
    et-0/0/35 {
        unit 0 {
```

```
            family ethernet-switching {
                interface-mode trunk;
                vlan {
                    members all;
                }
            }
        }
    }
}
```

And now you need to configure a similar redundancy group configuration on AD-2. For brevity, let's simply show the resultant configuration and not the individual set statements. Take note that although the redundancy group ID is the same, the Chassis-ID is different:

AD-2

```
chassis {
    satellite-management {
        redundancy-groups {
            chassis-id 2;
            dc-pod-1 {
                redundancy-group-id 1;
                peer-chassis-id 1 {
                    inter-chassis-link et-0/0/35;
                    liveness-detection {
                        minimum-interval 2000;
                        multiplier 3;
                        transmit-interval {
                            minimum-interval 2000;
                        }
                    }
                }
                satellite all;
            }
        }
    }
}
interfaces {
    et-0/0/35 {
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan {
                    members all;
                }
            }
        }
    }
}
```

Be sure to commit the configuration on AD-2 before moving onto the next section. From here on out, all future configuration will be taking place on only one of the aggregation devices as everything will take place within the config group and therefore get synchronized between the peers.

Configuration Groups and Apply Groups

Now that commit synchronization is enabled and there is a redundancy group, the last piece of the puzzle is to configure a configuration group for common elements that need to be shared amongst all of the aggregation devices. In a typical Junos Fusion Data Center architecture, this means putting all the relevant information regarding satellites into a config group – enabling certain ports to be cascade ports, assigning the cascade ports to FPCs, and configuring our satellite upgrade group membership and any other relevant satellite management options.

NOTE When initially cabling up the satellites to redundant aggregation devices, it's important to take care to keep the cascade ports the same on both aggregation devices. A big part of the simplicity to be gained by utilizing configuration groups and apply groups can only be recognized if the configuration can be mirrored on both devices – this can only be truly achieved if the same cascade ports on both aggregation devices are going to a given satellite. For example, assuming et-0/0/1 on AD-1 is connected to Satellite 101, you want to ensure that et-0/0/1 on AD-2 is also connected to Satellite 101 as well.

Let's first start by creating the configuration group, in this case, aptly entitled `fusion-config-group`:

```
{master:0}[edit]
jedi@AD-1# set groups fusion-config-group when peers 172.16.110.24

{master:0}[edit]
jedi@AD-1# set groups fusion-config-group when peers 172.16.110.25
```

Now you need to add additional information regarding the satellites and other relevant commands. In our case, since we previously configured some of these things under the [edit chassis satellite-management] stanza in our single aggregation device architecture, the easiest thing would be to move these elements to a configuration group so they can be replicated between both aggregation devices. We will then evaluate adding additional satellites so that the proper methodology will become evident when dealing with a dual aggregation device architecture.

You can simply copy and paste the existing configuration elements that you want to move to the configuration group, and then remove anything that is moved to the configuration group from the [edit chassis satellite-management] stanza. If you look at the current configuration under this stanza you'll see that everything can be moved except for the redundancy group configuration:

AD-1

```
chassis {
    satellite-management {
        fpc 100 {
            cascade-ports et-0/0/0;
```

```
        }
        fpc 101 {
            cascade-ports et-0/0/1;
        }
        fpc 102 {
            cascade-ports et-0/0/2;
        }
        redundancy-groups {
            chassis-id 1;
            dc-pod-1 {
                redundancy-group-id 1;
                peer-chassis-id 2 {
                    inter-chassis-link et-0/0/35;
                    liveness-detection {
                        minimum-interval 2000;
                        multiplier 3;
                        transmit-interval {
                            minimum-interval 2000;
                        }
                    }
                }
                satellite all;
            }
        }
        upgrade-groups {
            upgrade-group-1 {
                satellite [ 100 102 ];
            }
            upgrade-group-2 {
                satellite 101;
            }
        }
        auto-satellite-conversion {
            satellite 102;
        }
    }
}
```

Let's start by copying and pasting the FPCs, as well as the upgrade groups and auto-satellite-conversion configuration elements into a configuration group using the Junos `load merge terminal relative` command:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# load merge terminal relative
[Type ^D at a new line to end input]
chassis {
    satellite-management {
        fpc 100 {
            cascade-ports et-0/0/0;
        }
        fpc 101 {
            cascade-ports et-0/0/1;
        }
        fpc 102 {
            cascade-ports et-0/0/2;
        }
        upgrade-groups {
            upgrade-group-1 {
```

```
                satellite [ 100 102 ];
            }
            upgrade-group-2 {
                satellite 101;
            }
        }
        auto-satellite-conversion {
            satellite 102;
        }
    }
}
load complete
```

You also need to take care to copy the interface configuration for the cascade ports from the [interfaces] stanza:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# load merge terminal relative
[Type ^D at a new line to end input]
interfaces {
    et-0/0/0 {
        cascade-port;
    }
    et-0/0/1 {
        cascade-port;
    }
    et-0/0/2 {
        cascade-port;
    }
}
load complete
```

Now that we've copied over our config bits, let's take a final look at the configuration group:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# show
when {
    peers [ 172.16.110.24 172.16.110.25 ];
}
chassis {
    satellite-management {
        fpc 100 {
            cascade-ports et-0/0/0;
        }
        fpc 101 {
            cascade-ports et-0/0/1;
        }
        fpc 102 {
            cascade-ports et-0/0/2;
        }
        upgrade-groups {
            upgrade-group-1 {
                satellite [ 100 102 ];
            }
            upgrade-group-2 {
                satellite 101;
```

```
            }
        }
        auto-satellite-conversion {
            satellite 102;
        }
    }
}
interfaces {
    et-0/0/0 {
        cascade-port;
    }
    et-0/0/1 {
        cascade-port;
    }
    et-0/0/2 {
        cascade-port;
    }
}
```

As everything appears to be correct, you can now delete these elements from the global configuration as these elements are now present within the configuration group. From here on out, all future satellite additions or changes will take place from within the configuration group:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# top

{master:0}[edit]
jedi@AD-1# delete interfaces et-0/0/0

{master:0}[edit]
jedi@AD-1# delete interfaces et-0/0/1

{master:0}[edit]
jedi@AD-1# delete interfaces et-0/0/2

{master:0}[edit]
jedi@AD-1# delete chassis satellite-management fpc 100

{master:0}[edit]
jedi@AD-1# delete chassis satellite-management fpc 101

{master:0}[edit]
jedi@AD-1# delete chassis satellite-management fpc 102

{master:0}[edit]
jedi@AD-1# delete chassis satellite-management upgrade-groups

{master:0}[edit]
jedi@AD-1# delete chassis satellite-management auto-satellite-conversion
```

The only configuration elements that now exist under the [edit chassis] level of the hierarchy are as follows:

```
jedi@AD-1# show chassis
satellite-management {
    redundancy-groups {
```

```
        chassis-id 1;
        dc-pod-1 {
            redundancy-group-id 1;
            peer-chassis-id 2 {
                inter-chassis-link et-0/0/35;
                liveness-detection {
                    minimum-interval 2000;
                    multiplier 3;
                    transmit-interval {
                        minimum-interval 2000;
                    }
                }
            }
            satellite all;
        }
    }
}
```

The last piece of the puzzle is to take the configuration group and apply it as an `apply-group` at the root of the hierarchy:

```
{master:0}[edit]
jedi@AD-1# set apply-groups fusion-config-group
```

Commit our configuration and with a little luck we should expect to see our `fusion-config-group` elements synchronized to AD-2:

```
{master:0}[edit]
jedi@AD-1# commit
configuration check succeeds
commit complete
```

Let's take a look at AD-2 to ensure that it has received the relevant configuration elements as a result of the commit synchronization and the applied configuration groups:

```
jedi@AD-2# show groups
fusion-config-group {
    when {
        peers [ 172.16.110.24 172.16.110.25 ];
    }
    chassis {
        satellite-management {
            fpc 100 {
                cascade-ports et-0/0/0;
            }
            fpc 101 {
                cascade-ports et-0/0/1;
            }
            fpc 102 {
                cascade-ports et-0/0/2;
            }
            upgrade-groups {
                upgrade-group-1 {
                    satellite [ 100 102 ];
                }
```

```
                upgrade-group-2 {
                    satellite 101;
                }
            }
            auto-satellite-conversion {
                satellite 102;
            }
        }
    }
    interfaces {
        et-0/0/0 {
            cascade-port;
        }
        et-0/0/1 {
            cascade-port;
        }
        et-0/0/2 {
            cascade-port;
        }
    }
}
```

And let's verify:

```
{master:0}
jedi@AD-2> show chassis satellite
                         Device          Cascade      Port       Extended Ports
Alias            Slot    State           Ports        State      Total/Up
_sd100           100     Online          et-0/0/0     online     15/3
                                         et-0/0/35*   backup
_sd101           101     Online          et-0/0/1     online     18/3
                                         et-0/0/35*   backup
_sd102           102     Online          et-0/0/2     online     2/2
                                         et-0/0/35*   backup
```

You can see clearly that despite not having configured any cascade ports or satellites manually on AD-2, there are indeed satellites connected. Not only are they directly connected via the local cascade ports defined on AD-2, but they are also reachable through a backup path via AD-1 on the et-0/0/35 interface which was the inter-chassis link defined in our redundancy group configuration:

Now that AD-2 is properly synchronized and has established relationships with the satellite devices, let's switch back to AD-1 to verify additional device state. The following commands are used to display ICCP status as well as the status of our Redundancy Group configuration:

```
{master:0}
jedi@AD-1> show chassis satellite redundancy-group

            Cluster          Peer          Peer             Local         Device
Name        State            Chassis ID    Chassis SN       Chassis ID    Count
dc-pod-1    Online           2             DC491            1             3/3/3

{master:0}
jedi@AD-1> show iccp
```

```
Redundancy Group Information for peer 10.0.0.2
  TCP Connection       : Established
  Liveliness Detection : Up
  Redundancy Group ID          Status
    1                           Up

Client Application: l2ald_iccpd_client
  Redundancy Group IDs Joined: None

Client Application: MCSNOOPD
  Redundancy Group IDs Joined: None
```

You can also see the status of various FPC slots that are associated with the redundancy group configuration:

```
{master:0}
jedi@AD-1> show chassis satellite redundancy-group devices

Cluster                                        Local          Peer
name           Slot-ID          State          State
dc-pod-1       100              online         online
dc-pod-1       101              online         online
dc-pod-1       102              online         online
```

Adding Satellites to a Dual Aggregation Device Topology

Let's wrap this section up with how to add additional satellites to the Junos Fusion Data Center topology once moved to a dual aggregation device topology. As should be clear by now, instead of making changes under the [edit chassis satellite-management] stanza of the hierarchy, you will be making changes to the configuration group.

Here we'll add satellite 103 so it is reachable on ge-0/0/3 and add this satellite to the software upgrade group `upgrade-group-2` and enable `auto-satellite-conversion`:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces et-0/0/3 cascade-port

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set chassis satellite-management fpc 103 cascade-ports et-0/0/3

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set chassis satellite-management upgrade-groups upgrade-group-2 satellite 103

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set chassis satellite-management auto-satellite-conversion satellite 103
```

Now commit the configuration and watch the satellite connect to both aggregation devices:

```
{master:0}[edit]
jedi@AD-1# commit and-quit
configuration check succeeds
commit complete
Exiting configuration mode
```

```
{master:0}
jedi@AD-1> show chassis satellite
                        Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    Online          et-0/0/0     online     15/3
                                        et-0/0/35*   backup
_sd101           101    Online          et-0/0/1     online     18/3
                                        et-0/0/35*   backup
_sd102           102    Online          et-0/0/2     online     2/2
                                        et-0/0/35*   backup
_sd103           103    ModeChanging    et-0/0/3     present
```

After a few minutes, once the aggregation device has pushed down the satellite software to the newly added satellite, it is fully part of the Junos Fusion Data Center:

```
{master:0}
jedi@AD-1> show chassis satellite
                        Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    Online          et-0/0/0     online     15/3
                                        et-0/0/35*   backup
_sd101           101    Online          et-0/0/1     online     18/3
                                        et-0/0/35*   backup
_sd102           102    Online          et-0/0/2     online     2/2
                                        et-0/0/35*   backup
_sd103           103    Online          et-0/0/3     online     2/2
                                        et-0/0/35*   backup
```

Special Considerations for EX4300 Satellites

A Junos Fusion Data Center wouldn't be complete without adding at least one EX4300 satellite to the mix. However, in testing, additional configuration was required beyond what's stated in the technical documentation in order to properly get an EX4300 satellite connected into the Junos Fusion Data Center. This may or may not be the case in your environment, but take note of some of the additional settings here in the event you have difficulty attaching an EX4300 as a satellite.

First cable the EX4300 satellites into the aggregation devices using the 40Gbps QSFP+ ports as uplinks. The documentation starts by stating that all satellites need to start on a clean slate, once they are running the appropriate version of Junos. So first clear all resident and lingering configuration on the EX4300 by issuing the `request system zeroize` command:

```
{master:0}
root> request system zeroize
warning: System will be rebooted and may not boot without configuration
Erase all data, including configuration and log files? [yes,no] (no) yes

warning: ipsec-key-management subsystem not running - not needed by configuration.
warning: zeroizing fpc0
```

At this point you must modify the 40Gbps QSFP+ Virtual Chassis ports (VCP) on an EX 4300 to convert them to regular network ports. Before you convert them, check to see that these interfaces appear as VCP interfaces:

```
{master:0}
root> show interfaces terse | match vcp-
vcp-255/1/0             up    down
vcp-255/1/0.32768       up    down
vcp-255/1/1             up    down
vcp-255/1/1.32768       up    down
```

You can easily convert these ports by issuing the `request virtual-chassis vc-port delete` command:

```
{master:0}
root> request virtual-chassis vc-port delete pic-slot 1 port 0
vc-port successfully deleted

{master:0}
root> request virtual-chassis vc-port delete pic-slot 1 port 1
```

Once you have converted these ports, you can check to see that they no longer appear as VCP interfaces but are now listed as standard `et-` style network interfaces:

```
{master:0}
root> show interfaces terse | match vcp-

{master:0}
root> show interfaces terse | match et-
et-0/1/0                up    down
et-0/1/0.16386          up    down
et-0/1/1                up    down
et-0/1/1.16386          up    down
```

However, notice from the output that the Link Status is showing that the et-0/1/0 and et-0/1/1 interfaces are `down`. As long as these interfaces are down, there is no way they are going to be able to communicate with the aggregation devices and therefore will be unable to become satellites in the Junos Fusion Data Center. In order to force these interfaces into an `up` state so they may communicate with the aggregation devices, you may need to issue the following commands on the EX4300 satellite:

```
{master:0}[edit]
root# set system root-authentication plain-text-password
New password:
Retype new password:

{master:0}[edit]
root# set interfaces et-0/1/0 ether-options no-auto-negotiation

{master:0}[edit]
root# set interfaces et-0/1/1 ether-options no-auto-negotiation

{master:0}[edit]
root# commit and-quit
```

```
configuration check succeeds
commit complete
Exiting configuration mode
```

Once this configuration was committed, you can see that the QSFP ports are listed in an up state:

```
root> show interfaces terse | match et-
et-0/1/0                up    up
et-0/1/0.16386          up    up
et-0/1/1                up    up
et-0/1/1.16386          up    up
```

Much better. Okay, at this point let's proceed with the rest of the satellite installation from the aggregation device and connect to this satellite on cascade port et-0/0/4 and assign it to FPC slot 104. In addition, add it to the satellite upgrade group upgrade-group-1 and then enable auto satellite conversion:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set chassis satellite-management fpc 104 cascade-ports et-0/0/4

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set chassis satellite-management upgrade-groups upgrade-group-1 satellite 104

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set chassis satellite-management auto-satellite-conversion satellite 104
```

Let's commit the configuration and verify that you can see the satellite connecting to both aggregation devices:

```
{master:0}[edit]
jedi@AD-1# commit and-quit
configuration check succeeds
commit complete
Exiting configuration mode

{master:0}
jedi@AD-1> show chassis satellite
                         Device          Cascade      Port       Extended Ports
Alias            Slot    State           Ports        State      Total/Up
_sd100           100     Online          et-0/0/0     online     15/3
                                         et-0/0/35*+  backup
_sd101           101     Online          et-0/0/1     online     18/3
                                         et-0/0/35*+  backup
_sd102           102     Online          et-0/0/2     online     2/2
                                         et-0/0/35*+  backup
_sd103           103     Online          et-0/0/3     online     2/2
                                         et-0/0/35*+  backup
_sd104           104     ModeChanging    et-0/0/4     online     50/5
                                         et-0/0/35*   backup
```

After a few minutes, once the aggregation device has pushed down the satellite software to the newly added satellite, you can see it is fully part of the Junos Fusion Data Center:

```
{master:0}
jedi@AD-1> show chassis satellite
                        Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    Online          et-0/0/0     online     15/3
                                        et-0/0/35*+  backup
_sd101           101    Online          et-0/0/1     online     18/3
                                        et-0/0/35*+  backup
_sd102           102    Online          et-0/0/2     online     2/2
                                        et-0/0/35*+  backup
_sd103           103    Online          et-0/0/3     online     2/2
                                        et-0/0/35*+  backup
_sd104           104    Online          et-0/0/4     online     50/5
                                        et-0/0/35*   backup
```

Easy. Let's move on. We're already halfway through this *Day One* book.

Chapter 4

Junos Fusion Data Center Administration

This chapter tackles some of the basic administrative functions within a typical Junos Fusion Data Center. Once you get the topology configured with the dual aggregation devices and multiple satellites it's time to turn to the day-to-day operations – including such functions as verifying the status of satellites, software upgrade groups, port status, assigning aliases to our satellites, and lastly, how to remove a satellite and revert a satellite back to Junos should you wish to take it out of the system. Looking ahead, Chapter 5 will examine adding hosts into the Junos Fusion Data Center.

Verifying Satellite Device Status

Once you have your Junos Fusion Data Center up and running, you need to keep it up and running by looking at the commands and common procedures that allow verification of satellite device states, port states, satellite software versions, and commands that display the operational state of Junos Fusion. Experience has shown these are some of the most useful commands when working within the Junos Fusion architecture.

Verification of Satellite Configuration and Connectivity

By far, the best command in the whole arsenal is the `show chassis satellite` command. It's been demonstrated many times throughout this book, because it's important to call attention to the various aspects being displayed:

```
{master:0}
jedi@AD-1> show chassis satellite
                        Device          Cascade      Port       Extended Ports
Alias           Slot    State           Ports        State      Total/Up
```

```
_sd100           100    Online          et-0/0/0     online     15/3
                                        et-0/0/35*+  backup
_sd101           101    Online          et-0/0/1     online     18/3
                                        et-0/0/35*+  backup
_sd102           102    Online          et-0/0/2     online     2/2
                                        et-0/0/35*+  backup
_sd103           103    Online          et-0/0/3     online     2/2
                                        et-0/0/35*+  backup
_sd104           104    Online          et-0/0/4     online     50/5
                                        et-0/0/35*   backup
```

From the output of the the `show chassis satellite` command, you can see a wealth of information. You can clearly see the satellites that are being recognized by the aggregation device, and you can see the status of the satellite and the respective cascade ports going down to those satellites. You also have an indication that a backup path is available to reach the satellite via the et-0/0/35 ICL interface via AD-2. In addition, there's an indicator of the total number of extended ports that are available via a satellite. That's a lot of great information.

Verification of Satellite Device Hardware Model

The `show chassis satellite terse` command can be used to verify the hardware model of each satellite device. In addition, you can see which version of the satellite network operating system is installed on each satellite:

```
{master:0}
jedi@AD-1> show chassis satellite terse
       Device                          Extended Ports
Slot   State           Model           Total/Up    Version
100    Online          QFX5100-48S-6Q  15/3        3.1R1.4
101    Online          QFX5100-48S-6Q  18/3        3.1R1.4
102    Online          QFX5100-24Q-2P  2/2         3.1R1.4
103    Online          QFX5100-24Q-2P  2/2         3.1R1.4
104    Online          EX4300-48T      50/5        3.2R1.1
```

Verification of Cascade and Uplink Ports

The `show chassis satellite interface` command can be used to verify the status of both cascade and uplink ports. In addition, you can see the ICL link and any LAG interfaces that have been assigned to the satellites as extended ports:

```
{master:0}
jedi@AD-1> show chassis satellite interface

Interface         State       Type        DF-Role
et-0/0/0          Up          Cascade     NA
et-0/0/1          Up          Cascade     NA
et-0/0/2          Up          Cascade     NA
et-0/0/3          Up          Cascade     NA
et-0/0/35         Up          ICL         NA
et-0/0/4          Up          Cascade     NA
lo0               Up          Loopback    NA
sd-100/0/0        Up          Satellite   NON-DF
```

```
sd-101/0/0        Up           Satellite    NON-DF
sd-102/0/0        Up           Satellite    NON-DF
sd-103/0/0        Up           Satellite    NON-DF
sd-104/0/0        Up           Satellite    NON-DF
ae0               Up
```

Verification of Satellite Neighbor State

The `show chassis satellite neighbor` command is used to give additional information regarding the state of a given satellite device. The `Two-Way` state is an indicator that the satellite is properly initialized and a two-way relationship has been formed, meaning that traffic can be sent and received over the cascade port to the satellite. In addition, you can also see the hardware model and the satellite network operating system software version that is deployed on any given satellite:

```
{master:0}
jedi@AD-1> show chassis satellite neighbor
Interface   State      Port Info   System Name  Model           SW Version
et-0/0/4    Two-Way    et-0/1/0    sd104        EX4300-48T      3.2R1.1
et-0/0/3    Two-Way    et-0/0/0    sd103        QFX5100-24Q-2P  3.1R1.4
et-0/0/2    Two-Way    et-0/0/0    sd102        QFX5100-24Q-2P  3.1R1.4
et-0/0/1    Two-Way    et-0/0/48   sd101        QFX5100-48S-6Q  3.1R1.4
et-0/0/0    Two-Way    et-0/0/48   sd100        QFX5100-48S-6Q  3.1R1.4
```

Viewing Satellite Log Information

One of the most useful troubleshooting methods beyond using traceoptions is to utilize the `show chassis satellite` command paired with `fpc-slot` and the `extensive` knobs. This constrains the output to a given satellite associated with a certain FPC slot, and gives you an action log that is incredibly useful for detailed analysis:

```
{master:0}
jedi@AD-1> show chassis satellite fpc-slot 102 extensive
                         Device          Cascade      Port       Extended
Alias            Slot    State           Ports        State      Ports
_sd102           102     Online          et-0/0/2     online     2/2
                                         et-0/0/35*+  backup

When                 Event                          Action
Oct 23 16:05:38.70   Tx slot dependency info
Oct 23 16:05:37.201  Uplink ready RX                Uplink: et-102/0/0
                                                    First uplink UP
                                                    Update anchor NH
                                                    Cascade-Port et-0/0/2, Uplink port et-
102/0/0, ifl-idx 563
                                                    Local data path UP
                                                    Data path reachable.
                                                    Tx uplink-rx-ready to SD for port: et-102/0/0
Oct 23 16:05:37.201  csp-rx-sync-msg                CSP device state: Online
Oct 23 16:05:29.489  SW Version In-Sync
Oct 23 16:05:29.489  Rx SW-Version JSON-RPC response 3.1R1.4
                                                    new sw-update FSM state: version-in-sync
Oct 23 16:05:29.485  JSON-RPC session UP           new sw-update FSM state: version-req-rsp-pending
Oct 23 16:05:29.436  csp-rx-open-msg               CSP device state: SyncWait
```

```
                                                TX CSP-D Host Name Request
                                                CSP version: 0
                                                UFD supported, version: 1
                                                CoS supported, version: 1
                                                AE supported, version: 0
                                                Uplink port pinning supported, version: 0
                                                Local port mirroring supported, version: 0
Oct 23 16:05:29.432  csp-session-up            CSP device state: OpenWait
Oct 23 16:05:28.508  Transition timer expires  New connectivity state: Local
                                                Delete rt 172.16.0.102/32 next-hop: et-0/0/35
                                                Add rt 172.16.0.102/32 next-hop: et-0/0/2
Oct 23 16:05:28.497  Refresh color for slot    Device reachable via remote AD
                                                Set color for device
Oct 23 16:05:28.482  csp-notif-lldp-session-up CSP device state: ProvSessionDn
Oct 23 16:05:28.482  LLDP adjacency UP         SD rebooted
                                                Provisioning PID: 3011 PFE PID: 3029
                                                dev resolve state: Rslv-Active
                                                icl resolve state: Rslv-Complete
                                                Set color for device
                                                Set color operation fails
                                                New connectivity state: ICL
                                                Add rt 172.16.0.102/32 next-hop: et-0/0/35
                                                New device state: Online
                                                if resolve state: Rslv-Complete
                                                Uplink not ready for data path
                                                New connectivity state: Local-Sw-Wait
                                                IF: et-0/0/2, add a slot-id TLV: 102
Oct 23 16:05:27.596  Peer Chassis advertised   dev resolve state: Rslv-Pending
                                                On peer AD data path DN
                                                icl resolve state: Rslv-Pending
```

Verification of Extended Port Status

The `show chassis satellite extended-port` command can be used to verify the status of available extended ports in your Junos Fusion Data Center in addition to displaying the status of both the receive (Rx) path as well as the transmit (Tx) path. A state of `AddComplete` is an indicator that this port has been added as an extended port into the Junos Fusion Data Center topology.

```
{master:0}
jedi@AD-1> show chassis satellite extended-port
Legend for interface types:
   * -- Uplink interface
   + -- Clustering interface
                              Rx             Tx             Admin/Op IFD
Name           State          Request State  Request State  State    Idx   PCID
et-100/0/48*   AddComplete    None           Ready          Up/Up    183   150
et-100/0/49    AddComplete    None           Ready          Up/Up    184   154
et-101/0/48*   AddComplete    None           Ready          Up/Up    168   150
et-101/0/49    AddComplete    None           Ready          Up/Up    169   154
et-102/0/0*    AddComplete    None           Ready          Up/Up    187   102
et-102/0/1     AddComplete    None           Ready          Up/Up    188   106
et-103/0/0*    AddComplete    None           Ready          Up/Up    185   102
et-103/0/1     AddComplete    None           Ready          Up/Up    186   106
et-104/1/0*    AddComplete    None           Ready          Up/Up    238   155
et-104/1/1     AddComplete    None           Ready          Up/Up    239   156
ge-100/0/0     AddComplete    None           Ready          Up/Up    170   102
<--- output truncated for brevity -à
```

Verification of Satellite Software Versions

The `show chassis satellite software` command can be used to verify the status of satellite software that reside on the aggregation device, as well as what software upgrade groups the software is associated with. From the output below, you can see that the satellite network operating system software version 3.1R1.4 has been installed on the aggregation device and is currently associated with both upgrade-group-1 as well as upgrade-group-2:

```
{master:0}
jedi@AD-1> show chassis satellite software
Version                         Platforms                       Group
3.1R1.4                         i386 ppc arm arm563xx           upgrade-group-1
                                                                upgrade-group-2
```

Verification of Satellite Software Upgrade Group

The `show chassis satellite upgrade-group detail` command can be used to verify the software upgrade groups that have been configured in a Junos Fusion Data Center. In addition, it shows which satellites are part of a given software upgrade group and whether the software installed on a particular satellite matches that of the software upgrade group. If these match, the device state will list as `version-in-sync`:

```
{master:0}
jedi@AD-1> show chassis satellite upgrade-group detail
Software upgrade group: upgrade-group-1
Software package version: 3.1R1.4
  Slot    Device State
  100     version-in-sync
  102     version-in-sync
  104     version-in-sync

Software upgrade group: upgrade-group-2
Software package version: 3.1R1.4
  Slot    Device State
  101     version-in-sync
  103     version-in-sync
```

Aliases

Aliases are labels that you can assign a satellite device to assist in identification. By default, satellites take an alias in the form of '_sd' followed by the FPC number, for example _sd100. Although this clearly labels the FPC slot that a particular satellite has been assigned to, it's not very useful beyond that. As an administrator, you can assign an alias of your choosing such that you can give your satellites more meaningful names. Although the alias is an optional configuration parameter, it can allow you to more clearly identify a particular satellite in your show command outputs. For example, perhaps you want to give the satellites an alias to indicate which rack they are installed in, or perhaps you'd like an indicator as to whether the satellite is top of rack, bottom of rack, or whatever creative distinction you'd like to make.

Let's look at the previously defined FPCs 100-104 and assign aliases to each of these. We'll designate the rack the satellite is installed in, the underlying model, and also its position in the rack: 'A' for top of rack and 'B' for bottom of rack:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set chassis satellite-management fpc 100 alias Rack1-QFX5100-A

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set chassis satellite-management fpc 101 alias Rack1-QFX5100-B

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set chassis satellite-management fpc 102 alias Rack2-QFX5100-A

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set chassis satellite-management fpc 103 alias Rack2-QFX5100-B

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set chassis satellite-management fpc 104 alias Rack3-EX4300-A

{master:0}[edit groups fusion-config-group]
jedi@AD-1# commit and-quit
configuration check succeeds
commit complete
Exiting configuration mode
```

Once committed, verify the result of the configuration change with one of this chapter's various `show` command outputs:

```
{master:0}
jedi@AD-1> show chassis satellite
                     Device          Cascade      Port       Extended Ports
Alias            Slot  State           Ports        State      Total/Up
Rack1-QFX5100-A  100   Online          et-0/0/0     online     15/3
                                       et-0/0/35*+  backup
Rack1-QFX5100-B  101   Online          et-0/0/1     online     18/3
                                       et-0/0/35*+  backup
Rack2-QFX5100-A  102   Online          et-0/0/2     online     2/2
                                       et-0/0/35*+  backup
Rack2-QFX5100-B  103   Online          et-0/0/3     online     2/2
                                       et-0/0/35*+  backup
Rack3-EX4300-A   104   Online          et-0/0/4     online     50/5
                                       et-0/0/35*   backup
```

Removing a Satellite and Reverting to Junos

In some cases, it may be useful to simply remove a satellite from service and have Junos installed on the device. This might be necessary if a satellite is to be decommissioned but intended for use elsewhere in the network where the features of full-blown Junos might be more useful.

The first thing you need to do is to disable automatic satellite conversion if it happens to be enabled for that satellite. This is important, because you don't want the aggregation devices attempting to convert the device back into a satellite and initiating a reinstall of the satellite network operating system software.

Once this is done, you will then convert the satellite to standalone Junos, but be aware there are some software requirements that must be met in order to proceed with a successful conversion:

- QFX5100 switches acting as satellite devices require a special version of Junos in order to be converted back into a standalone device running the Junos software. The version of Junos that must be installed is a Preboot eXecution Environment (PXE) version of Junos and includes 'pxe' in the package name, as can be seen in the output. QFX5100 must be PXE (discussed later in this section).
- The Junos version must be a version of Junos OS that supports conversion into a satellite device, in other words, Junos version 14.1X53-D43 or higher.
- If all else fails, you may need a "-signed" version of Junos.

Assuming the above requirements are met, let's attempt the removal of the satellite attached to FPC 103:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# delete chassis satellite-management auto-satellite-conversion satellite 103

{master:0}[edit groups fusion-config-group]
jedi@AD-1# commit and-quit
configuration check succeeds
commit complete
Exiting configuration mode
```

After commiting the configuration, you can then attempt to start the standalone Junos conversion process by initiating an install of Junos to the satellite device:

```
{master:0}
jedi@AD-1> request chassis satellite install /var/tmp/install-media-pxe-qfx-5-14.1X53-D45.3-
domestic-signed.tgz fpc-slot 103
Convert satellite device to Junos standalone device? [yes,no] (no) yes

Verified install-media-pxe-qfx-5-14.1X53-D45.3-domestic.
tgz signed by PackageProductionSwitchingEc_2015 method ECDSA256+SHA256
Initiating Junos standalone conversion on device 103...
Response from device:
  Conversion started
```

Once the command is issued, the satellite device stops participating in the Junos Fusion. After some time, you can see that the satellite device attached to FPC 103 is listed as `offline`:

```
{master:0}
jedi@AD-1> show chassis satellite fpc-slot 103
                        Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
Rack2-QFX5100-B  103    Offline        et-0/0/3     offline    2/0
```

And you can also look at the log data by using the **`extensive`** knob to reveal that the conversion has indeed been initiated:

```
{master:0}
jedi@AD-1> show chassis satellite fpc-slot 103 extensive
                        Device          Cascade      Port       Extended
Alias            Slot   State           Ports        State      Ports
Rack2-QFX5100-B  103    Offline         et-0/0/3     offline    2/0

When                 Event                          Action
Oct 24 10:44:33.431  Inactive timer expires
Oct 24 10:43:03.349  Peer Chassis withdraw           icl resolve state: Rslv-None
                                                    Clear color for device
                                                    Delete rt 172.16.0.103/32 next-hop: et-0/0/35
                                                    New connectivity state: None
                                                    New device state: Offline
                                                    dev resolve state: Rslv-Inactive
                                                    Start Aging timer (10 minutes)
Oct 24 10:43:03.309  Transition timer expires       New connectivity state: ICL
                                                    Add rt 172.16.0.103/32 next-hop: et-0/0/35
                                                    New device state: Online
Oct 24 10:43:03.284  csp-keepalive-expire           CSP device state: ProvSessionDn
Oct 24 10:43:03.283  LLDP adjacency lost            IF: et-0/0/3, delete slot-id TLV: 103
                                                    if resolve state: Rslv-None
                                                    Update anchor NH
                                                    Local data path DN
                                                    Data path not reachable. Uplink list empty.
                                                    Delete rt 172.16.0.103/32 next-hop: et-0/0/3
                                                    New connectivity state: ICL-Sw-Wait
                                                    New device state: Discovered
                                                    new sw-update FSM state: version-unknown
Oct 24 10:41:28.712  Rx SW-Update JSON-RPC response Conversion started
Oct 24 10:40:07.371  Start-SW-Update                Junos conversion
```

After some time, the device on FPC 103 will be converted and go through a few reboots during the conversion process. You can verify the conversion is complete by looking at the output of the `show chassis satellite` command:

```
{master:0}
jedi@AD-1> show chassis satellite fpc-slot 103
                        Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
Rack2-QFX5100-B  103    Standalone      et-0/0/3     present
```

The device shows but it is in a device state of `Standalone`. To make this device no longer appear in your `show chassis satellite` output, remove all references to it under the [`chassis satellite-management`] level of the hierarchy:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# delete chassis satellite-management fpc 103

{master:0}[edit groups fusion-config-group]
jedi@AD-1# delete chassis satellite-management upgrade-groups upgrade-group-2 satellite 103

{master:0}[edit groups fusion-config-group]
jedi@AD-1# delete interfaces et-0/0/3

{master:0}[edit groups fusion-config-group]
jedi@AD-1# commit and-quit
```

```
configuration check succeeds
commit complete
Exiting configuration mode
```

Verification reveals that the device is no longer listed as a satellite:

```
{master:0}
jedi@AD-1> show chassis satellite
                        Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
Rack1-QFX5100-A  100    Online          et-0/0/0     online     15/3
                                        et-0/0/35*+  backup
Rack1-QFX5100-B  101    Online          et-0/0/1     online     18/3
                                        et-0/0/35*+  backup
Rack2-QFX5100-A  102    Online          et-0/0/2     online     2/2
                                        et-0/0/35*+  backup
Rack3-EX4300-A   104    Online          et-0/0/4     online     50/5
                                        et-0/0/35*   backup
```

Chapter 5

Connecting to Hosts in Junos Fusion Data Center

This chapter completes the Junos Fusion Data Center configuration by connecting to hosts, which is, of course, the main purpose for building a Junos Fusion Data Center in the first place. Connecting to hosts represents the last milestone in the completed Junos Fusion Data Center, one that can span to thousands of hosts all managed via a single-pane-of-glass management.

The chapter shows you how to connect to hosts using some advanced features within Junos Fusion:

- Uplink Failure Detection
- Link Aggregation Groups (LAG) for connected hosts
- IRB interfaces on aggregation devices for Inter-VLAN routing
- VLAN autosensing and local switching

Uplink Failure Detection

Uplink failure detection (UFD) is a feature that can be enabled on a Junos Fusion Data Center to assist satellite devices in the detection of failure conditions on uplink interfaces towards aggregation devices.

When a satellite detects that an uplink towards an aggregation device is down and has uplink failure detection enabled, the satellite is able to shut down the extended ports connected to host devices. By tightly coupling the status of an uplink to the extended ports, downstream hosts are more quickly notified and can therefore adapt to the outage. Commonly, this is used when a host device is connected to more than one satellite device and can therefore adapt and reroute away from affected satellites and towards active satellites instead.

Uplink failure detection can be configured globally on all uplinks within a given Junos Fusion Data Center, or it can be configured individually at the satellite level. It can also be configured at both places simultaneously, however, as is typical in Junos, the more specific satellite device level configuration will take precedence over anything configured globally.

Let's start by enabling uplink failure detection globally, to all of the satellites using the default settings. The default settings specify that a minimum number of one active uplink must exist in order for the extended port to remain active. The defaults also enforce a holddown timer of six seconds, which is the amount of time that uplink failure detection must wait in order to enable a previously disabled extended port. The intention of a holddown timer is to prevent flapping of extended ports during transient uplink port states:

```
{master:0}[edit]
jedi@AD-1# set groups fusion-config-group chassis satellite-management uplink-failure-detection

{master:0}[edit]
jedi@AD-1# commit
configuration check succeeds
commit complete
```

After committing the configuration, use the `show chassis satellite detail fpc-slot fpc-slot-id-number` command to verify uplink failure detection operation and settings. In the sample shown here, uplink failure detection operation is confirmed for the satellite device using FPC slot 100:

```
{master:0}
jedi@AD-1> show chassis satellite detail fpc-slot 100
Satellite Alias: _sd100
FPC Slot: 100
Operational State: Online
Product Model: QFX5100-48S-6Q
Product Family: i386
Serial number: VF3715080147
Device Reachability: Local
System id: 88:a2:5e:59:9d:e0
Software package version: 3.1R1.4
Host software version: 3.0.3
Management Address: 172.16.0.100/32
Cascade interfaces:
    Interface Name: et-0/0/0 State: online
        Uplink Interface: et-100/0/48
        Adjacency state: Two-Way
        Last transition: 14:43:13
        Adjacency down count: 0
        Rx Packet: 5301 Last received packet: 00:00:10
        Peer adjacency information: 14:43:13
            Adjacency down count: 13
            Last down cause: Adjacency TLV missing
            SDPD restart detected: 3
Process information:
    Process Name: Provisioning PID: 3012 State: Running
        Number of restart detected: 0
```

```
        Uptime: 14:56:27
    Process Name: PFE PID: 3030 State: Running
        Number of restart detected: 0
        Uptime: 14:56:27
UFD config state: Enable (persist), Minimum link: 1,
Holdddown timer (seconds): 6
UFD operational state: Enable
Candidate uplink interfaces (pic/port):
        0/48
        0/49
        0/50
        0/51
        0/52
        0/53
```

NOTE The output here showing UFD configuration state and associated parameters is only visible once an equivalent configuration has been committed. When uplink failure detection is not enabled in the configuration, this information will not be present via the `show chassis satellite detail fpc-slot fpc-slot-id-number` command output.

Notice in the above output that the default parameters for "minimum-links" and "holddown" have been applied. This is because we simply enabled uplink failure detection with the default settings. Under normal circumstances this would be fine, but be aware that there are only certain scenarios where uplink failure detection would actually cause an extended port to go down, because most often a satellite will have redundant uplinks to multiple aggregation devices. A very simplistic scenario as shown in Figure 5.1, where each satellite only has a single uplink to a single aggregation device, and if that single uplink were to fail with uplink failure detection enabled, the extended port would then be marked as `down`:

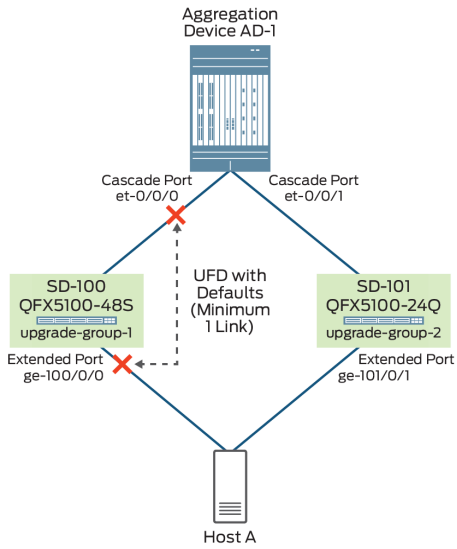

Figure 5.1 Single-Homed Satellite with UFD and Minimum-Links 1

In the real world, it would be more commonplace to have redundant uplinks to multiple aggregation devices, therefore, even if a given cascade port were down, the extended port will still be operational due to the fact that the extended port has a minimum link of one uplink available to it, via the alternate aggregation device. This is demonstrated in the topology used throughout this book, as shown in Figure 5.2.

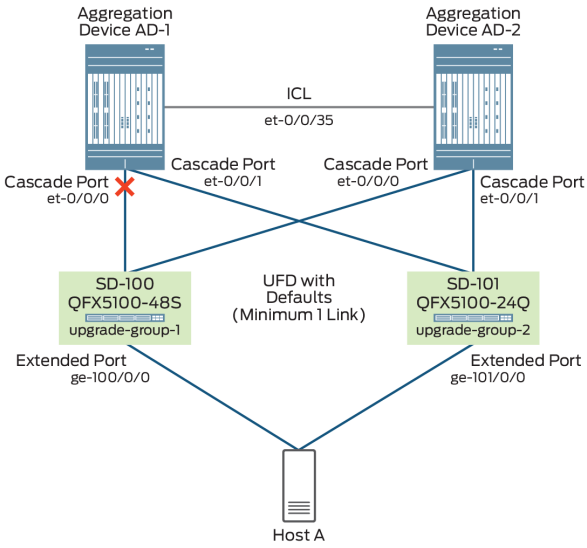

Figure 5.2 *Dual-Homed Satellites with UFD and Minimum-Links 1*

Let's examine this in more detail, using the default uplink failure detection parameters applied to the testbed topology shown here. You can see that even though the cascade port et-0/0/0 to satellite 100 is down, the extended ports on those satellites to the hosts are still showing as `up`:

```
{master:0}
jedi@AD-1> show interfaces et-0/0/0 terse
Interface               Admin Link Proto    Local                 Remote
et-0/0/0                up    down
et-0/0/0.32769          up    down inet     10.16.0.1/30
et-0/0/0.32770          up    down eth-switch

{master:0}
jedi@AD-1> show interfaces ge-100/0/0 terse
Interface               Admin Link Proto    Local                 Remote
ge-100/0/0              up    up
ge-100/0/0.0            up    up   aenet    --> ae0.0
```

In fact, you can see that even though the port state on the cascade port on AD-1 is `down`, the satellite on FPC 100 is still very much alive and well via its connection to AD-2, which is reachable via the ICL link et-0/0/35:

```
{master:0}
jedi@AD-1> show chassis satellite fpc-slot 100
                        Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    Online          et-0/0/0     down       15/2
                                        et-0/0/35*   online
```

There are circumstances where you may want to custom tailor the minimum link or holddown parameters to achieve a certain objective. For example, let's say you have a situation where a satellite is multihomed via 10GE to two aggregation devices for a total uplink of 20Gbps. If one of the cascade ports to that satellite were to fail, you would have only one available uplink for a total of 10Gbps. If this were undesirable and you wanted to signal to the extended ports that they should shut down so that the affected hosts would become aware of the situation, and reroute accordingly, you could accomplish this via a candidate uplink port policy.

Let's examine what happens in our testbed topology after a cascade port goes down with a candidate uplink port policy applied, mandating that a minimum link of two uplinks is required. Ideally, the extended ports should go down, as illustrated in Figure 5.3.

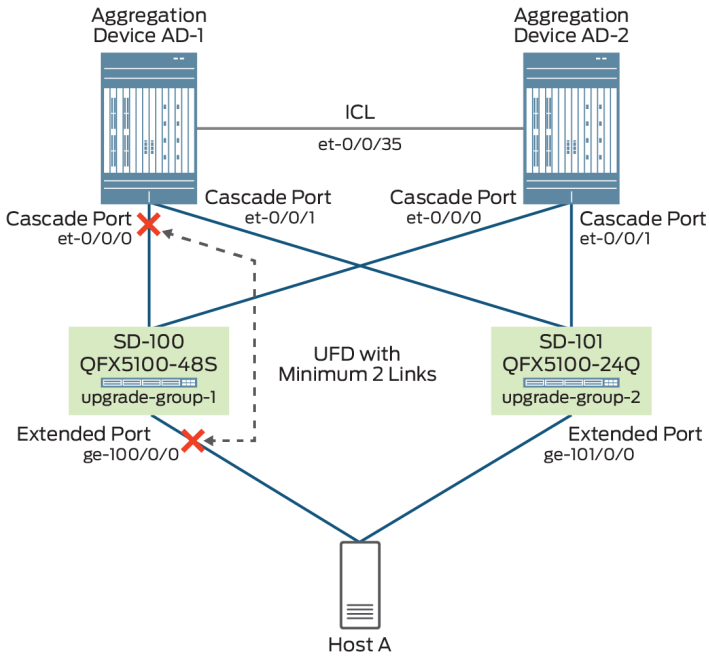

Figure 5.3 Dual-Homed Satellites with UFD and Minimum-Links 2

Let's apply a `candidate-uplink-port-policy` forcing a minimum of two uplinks on satellite 100 and then examine what happens when even just a single link fails:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set policy-options satellite-policies candidate-uplink-port-policy 2-link-policy minimum-
links 2

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set chassis satellite-management fpc 100 uplink-failure-detection candidate-uplink-
policy 2-link-policy

{master:0}[edit groups fusion-config-group]
jedi@AD-1# commit
configuration check succeeds
commit complete
```

Now let's examine to see if the resultant policy has been applied to satellite 100:

```
{master:0}
jedi@AD-1> show chassis satellite fpc-slot 100 detail
Satellite Alias: _sd100
FPC Slot: 100
Operational State: Online
Product Model: QFX5100-48S-6Q
Product Family: i386
Serial number: VF3715080147
Device Reachability: Local
System id: 88:a2:5e:59:9d:e0
Software package version: 3.1R1.4
Host software version: 3.0.3
Management Address: 172.16.0.100/32
Cascade interfaces:
    Interface Name: et-0/0/0 State: online
        Uplink Interface: et-100/0/48
        Adjacency state: Two-Way
        Last transition: 00:07:03
        Adjacency down count: 2
        Last down cause: Interface Down (last pkt rx 8 sec earlier)
        Rx Packet: 23598 Last received packet: 00:00:03
        Peer adjacency information: 00:07:03
            Adjacency down count: 15
            Last down cause: Interface Down
            SDPD restart detected: 3
Process information:
    Process Name: Provisioning PID: 3012 State: Running
        Number of restart detected: 0
        Uptime: 2d 19:05:45
    Process Name: PFE PID: 3030 State: Running
        Number of restart detected: 0
        Uptime: 2d 19:05:45
UFD config state: Enable (persist), Minimum link: 2,
Holdddown timer (seconds): 6
UFD operational state: Enable
Candidate uplink interfaces (pic/port):
        0/48
        0/49
        0/50
        0/51
        0/52
        0/53
```

You can see that the policy has been applied. At the same time—and this is a good example of Junos's behavior— the most explicit configuration takes precedence, and the global setting of `uplink-failure-detection` applied to all the other satellites is being overshadowed by the more specific uplink failure detection configuration. Note how the other FPCs are still using the UFD default of one minimum link:

```
{master:0}
jedi@AD-1> show chassis satellite fpc-slot 101 detail | match UFD
UFD config state: Enable (persist), Minimum link: 1,
UFD operational state: Enable
```

Let's create a failure condition by bringing one of the cascade ports down to satellite 100 and observing the outcome. In this case, a simple cable being pulled from the et-0/0/0 cascade port on AD-1 should do the trick:

```
{master:0}
jedi@AD-1> run show interfaces et-0/0/0 terse
Interface               Admin Link Proto    Local                 Remote
et-0/0/0                up    down
et-0/0/0.32769          up    down inet     10.16.0.1/30
et-0/0/0.32770          up    down eth-switch

{master:0}
jedi@AD-1> show chassis satellite fpc-slot 100
                        Device          Cascade      Port       Extended Ports
Alias            Slot   State           Ports        State      Total/Up
_sd100           100    UFDDown         et-0/0/0     down       15/1
                                        et-0/0/35*   online
```

Notice that et-0/0/0 now registers a link `down` state. Additionally, the `show chassis satellite` command reveals that the satellite port state is listed as `down` for cascade port et-0/0/0, but perhaps more importantly, the device state is now listed as `UFD-Down`. This is a clear indicator that uplink failure detection is working as expected. Let's examine the outcome on our extended ports reachable on satellite 100:

```
{master:0}
jedi@AD-1> show interfaces ge-100/0/0 terse
Interface               Admin Link Proto    Local                 Remote
ge-100/0/0              up    down
ge-100/0/0.0            up    down aenet    --> ae0.0
```

You can clearly see from the `show` command that by configuring a candidate-uplink-policy, you can force a minimum number of links to be used and therefore ensure that if that policy is not met, in terms of the number of uplinks available, extended ports can be put into a `down` state. This will, in turn, notify end-hosts that they should not use that path for packet forwarding.

The resulting configuration is here:

```
groups {
    fusion-config-group {
        chassis {
            satellite-management {
```

```
                fpc 100 {
                    cascade-ports et-0/0/0;
                    uplink-failure-detection {
                        candidate-uplink-policy 2-link-policy;
                    }
                }
                uplink-failure-detection;
            }
        }
        policy-options {
            satellite-policies {
                candidate-uplink-port-policy 2-link-policy {
                    minimum-links 2;
                }
            }
        }
    }
}
```

Link Aggregation and LACP

In modern data centers, hosts are likely to have multiple uplinks to redundant servers. This often comes in the form of an approach known as *link aggregation* whereby these multiple Ethernet uplinks are combined in parallel such that increased throughput and a higher level of redundancy are achieved. Traffic is typically load-balanced across the various component Ethernet member interfaces of an aggregated interface. In Junos, when you combine multiple Ethernet interfaces in this manner, it's called a *link aggregation group* (LAG).

As a LAG is comprised of multiple member links, even if a single member link fails, the LAG is still able to maintain connectivity over the remaining links. However, Junos Fusion Data Center goes above and beyond with an approach where a LAG could span across extended ports on multiple satellites. The benefit of configuring a LAG across multiple satellites is that you gain node-level redundancy as well as link-level redundancy. In other words, when LAG is used between a host and only a single satellite, if that satellite were to go down, the entire LAG would go down. By configuring a LAG across multiple satellites, you can still maintain connectivity through the LAG's remaining member interfaces through alternate satellites.

In addition to configuring LAG interfaces in Junos Fusion Data Center, you can also configure Link Aggregation Control Protocol (LACP) which provides additional functionality by allowing LAGs to identify communication failures and misconfigurations within a LAG. Misconfigurations are identified through the link monitoring capabilities of LACP, which can verify whether both ends of a particular LAG bundle are connected to the correct group.

Okay, let's take a look at configuring multiple extended ports in our Junos Fusion Data Center to support LAG. In the next example, we'll build a LAG using extended ports on two different satellite devices and we'll also configure LACP. Figure 5.4 illustrates what we want to achieve via the configuration.

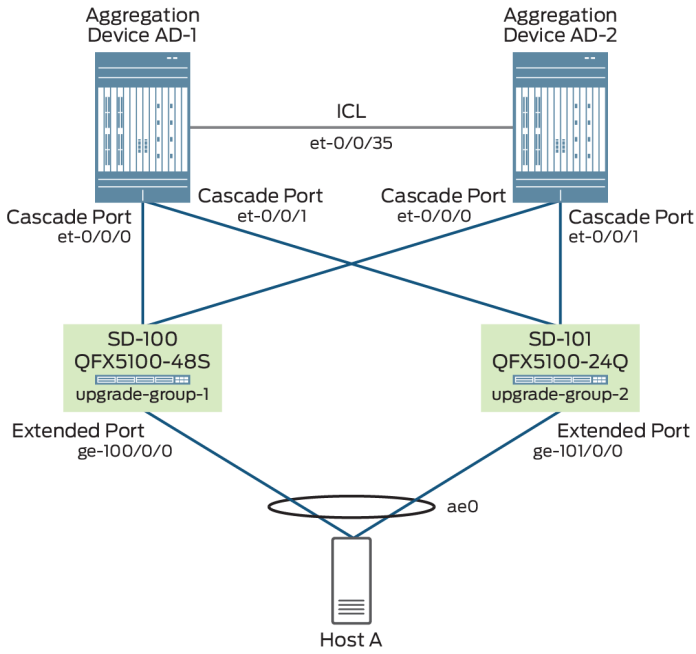

Figure 5.4 Link Aggregation and LACP

The first thing to do before you can configure a LAG is to enable a certain number of aggregated Ethernet interfaces on the box. Since these are by nature virtual interfaces, they do not exist by default. For this example, let's enable 100 aggregated Ethernet interfaces:

```
{master:0}[edit]
jedi@AD-1# set groups fusion-config-group chassis aggregated-devices ethernet device-count 100
```

Once the aggregated Ethernet interfaces have been created, you can create the LAG interface that will connect to the host, in this case ae0, which will connect to Host A:

```
{master:0}[edit]
jedi@AD-1# set groups fusion-config-group interfaces ae0 description "AE Interface to Host A"
```

Then assign the extended port interfaces to the aggregated Ethernet interface:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ge-100/0/0 ether-options 802.3ad ae0
jedi@AD-1# set interfaces ge-101/0/0 ether-options 802.3ad ae0
```

Finally, enable LACP on the aggregated Ethernet interface, where the extended ports on the satellite will act as the Active partner and it is expected that the host will be configured as the Passive partner.

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ae0 aggregated-ether-options lacp active
jedi@AD-1# set interfaces ae0 aggregated-ether-options lacp periodic fast
```

The last thing to do in order for the aggregated Ethernet interface to come up is to enable the `ethernet-switching` family on the interface:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ae0 unit 0 family ethernet-switching

{master:0}[edit groups fusion-config-group]
jedi@AD-1# commit
configuration check succeeds
commit complete
```

Once the configuration has been committed, verify that the LAG interface is operational and also that LACP is working as expected:

```
jedi@AD-1> show interfaces ae0 extensive
Physical interface: ae0, Enabled, Physical link is Up, Extended Port, multi-homed
  Interface index: 329, SNMP ifIndex: 643, Generation: 332
  Description: AE Interface to Host A
  Link-level type: Ethernet, MTU: 1514, Speed: 2Gbps, BPDU Error: None, Ethernet-
Switching Error: None,
  MAC-REWRITE Error: None, Loopback: Disabled, Source filtering: Disabled, Flow control: Disabled,
  Minimum links needed: 1, Minimum bandwidth needed: 1bps
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Current address: 00:31:46:ef:0a:56, Hardware address: 00:31:46:ef:0a:56
  Last flapped   : 2017-10-23 11:47:20 UTC (00:05:52 ago)
  Statistics last cleared: Never
  Extended port information:
    Satellite device port id  : 0
    Internal MC-AE id         : 3221225472
    Internal LACP port id     : 0
    Redundancy group          : 1
    Mode                      : Active Active
  Traffic statistics:
   Input  bytes  :             43809544                 1840 bps
   Output bytes  :             48610722                    0 bps
   Input  packets:               342919                    2 pps
   Output packets:               360942                    0 pps
   IPv6 transit statistics:
    Input  bytes  :                   0
    Output bytes  :                   0
    Input  packets:                   0
    Output packets:                   0
```

```
  Input errors:
    Errors: 0, Drops: 0, Framing errors: 0, Runts: 0, Giants: 0, Policed discards: 0, Resource errors: 0
  Output errors:
    Carrier transitions: 0, Errors: 0, Drops: 0, MTU errors: 0, Resource errors: 0
  Egress queues: 8 supported, 4 in use
  Queue counters:       Queued packets  Transmitted packets      Dropped packets
    0                                0                  601                    0
    3                                0                    0                    0
    4                                0                    0                    0
    7                                0                 1075                    0
  Queue number:         Mapped forwarding classes
    0                   best-effort
    3                   fcoe
    4                   no-loss
    7                   network-control

  Logical interface ae0.0 (Index 563) (SNMP ifIndex 644) (Generation 175)
    Flags: Up SNMP-Traps 0x24024000 Encapsulation: Ethernet-Bridge
    Statistics        Packets        pps         Bytes          bps
    Bundle:
        Input :             0          0             0            0
        Output:             0          0             0            0
    Adaptive Statistics:
        Adaptive Adjusts:          0
        Adaptive Scans  :          0
        Adaptive Updates:          0
    Link:
      ge-100/0/0.0
        Input :             0          0             0            0
        Output:             0          0             0            0
      ge-101/0/0.0
        Input :             0          0             0            0
        Output:             0          0             0            0

Aggregate member links: 2

  LACP info:       Role     System             System      Port     Port    Port
                          priority         identifier  priority   number     key
    ge-100/0/0.0   Actor       127  00:00:00:00:00:01       127     8962       1
    ge-100/0/0.0  Partner      127  4c:96:14:40:71:40       127        1       1
    ge-101/0/0.0   Actor       127  00:00:00:00:00:01       127     9218       1
    ge-101/0/0.0  Partner      127  4c:96:14:40:71:40       127        2       1
  LACP Statistics:       LACP Rx     LACP Tx   Unknown Rx   Illegal Rx
    ge-100/0/0.0             334         355            0            0
    ge-101/0/0.0              97         106            0            0
  Marker Statistics:   Marker Rx     Resp Tx   Unknown Rx   Illegal Rx
    ge-100/0/0.0               0           0            0            0
    ge-101/0/0.0               0           0            0            0
  Protocol eth-switch, MTU: 1514, Generation: 207, Route table: 6, Mesh Group: __all_ces__
```

At the top of the CLI output you can also see that the ae0 interfaces is an extended port and multihomed. This is a clear indicator that the member links are extended ports themselves and that they are multihomed across multiple satellite devices.

Finally, if you simply wish to look at LACP status without all the other information that is displayed via the `show interfaces` command, try using the `show lacp interfaces` command instead:

```
jedi@AD-1> show lacp interfaces
Aggregated interface: ae0
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      ge-100/0/0     Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      ge-100/0/0   Partner    No    No   Yes  Yes  Yes   Yes     Fast   Passive
      ge-101/0/0     Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      ge-101/0/0   Partner    No    No   Yes  Yes  Yes   Yes     Fast   Passive
    LACP protocol:        Receive State  Transmit State          Mux State
      ge-100/0/0                Current   Fast periodic Collecting distributing
      ge-101/0/0                Current   Fast periodic Collecting distributing
```

You can see that by using an aggregated Ethernet interface you can add extended ports as members that span across multiple satellites, in this case satellite 100 and satellite 101, all while gaining the additional benefits that LACP affords in identifying misconfigurations and communications issues. Good stuff.

The configuration is here:

```
groups {
    fusion-config-group {
        chassis {
            aggregated-devices {
                ethernet {
                    device-count 100;
                }
            }
        interfaces {
            ae0 {
                description “AE Interface to Host A”;
                aggregated-ether-options {
                    lacp {
                        active;
                        periodic fast;
                    }
                }
                unit 0 {
                    family ethernet-switching;
                }
            }
            ge-100/0/0 {
                ether-options {
                    802.3ad ae0;
                }
            }
            ge-101/0/0 {
                ether-options {
                    802.3ad ae0;
                }
            }
        }
    }
}
```

VLAN Configuration and IRB

Modern data centers most likely use VLANs to contain and isolate traffic between hosts at Layer 2. Hosts that need to speak with each other at Layer 2 are likely to be in the same subnet and are therefore perfect candidates for positioning within the same VLAN. At the same time, there are likely to be hosts that need to connect to each other that are not on the same VLAN, or hosts which may need reachability to external networks such as the Internet, and therefore Integrated Routing and Bridging (IRB) interfaces are likely to be used to route traffic at Layer 3.

The topology that we'll be using to build an example VLAN and IRB configuration is shown in Figure 5.5.

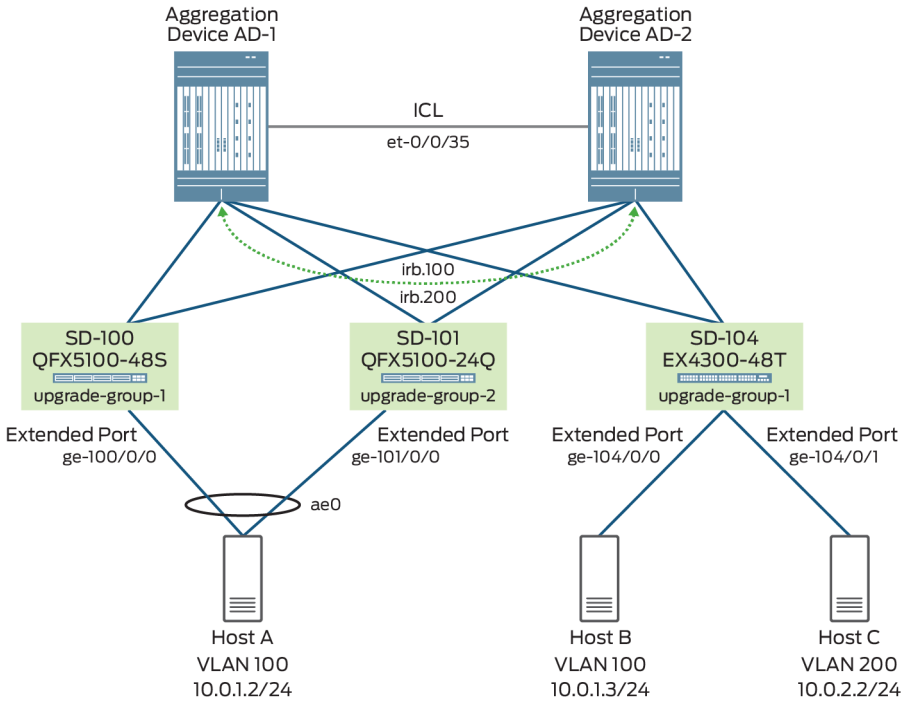

Figure 5.5 *VLAN Configuration and IRBs*

Figure 5.5 shows that we are building on the topology created in the previous example, with the added aggregated Ethernet interface ae0 to satellite 100 and 101. Now we'll be layering a VLAN 100 on this interface for Host A, and we'll also be extending that VLAN on satellite 104 extended port ge-104/0/0 to connect to Host B. An IRB interface, irb.100, will be added and will reside on both aggregation devices for any inter-VLAN or external routing purposes. In addition, we're adding a second VLAN, VLAN 200, on satellite 104 extended port ge-104/0/1, to connect to Host C with an IRB interface irb.200 added for inter-VLAN routing. The topology should demonstrate intra-VLAN switching as well as inter-VLAN routing.

Step one in the VLAN configuration is to assign the VLANs to the various extended ports:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ae0 unit 0 family ethernet-switching vlan members 100

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ge-104/0/0 description "GigE to Host B"

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ge-104/0/0 unit 0 family ethernet-switching vlan members 100

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ge-104/0/1 description "GigE to Host C"

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ge-104/0/1 unit 0 family ethernet-switching vlan members 200
```

The next step is to create the VLANs and number them.

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set vlans vlan100 vlan-id 100

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set vlans vlan200 vlan-id 200
```

Next configure the IRB interfaces that will be providing Layer 3 connectivity, whether for external routing purposes or for inter-VLAN routing:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces irb unit 100 family inet address 10.0.1.1/24

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces irb unit 200 family inet address 10.0.2.1/24
```

NOTE Note that here we have configured the unit number on the IRB interface to match that of the VLAN we intend to associate it with. Although the IRB unit number is an arbitrary number, and can be assigned any value, it is considered best practice to number this the same as the VLAN for which it provides routing services.

Finally, you must bind the IRB interfaces to the respective VLANs they will be providing services for, and we will be enabling MAC synchronization for. This is required in the case where multiple aggregation devices are sharing a common IRB and must be enabled in order to ensure both devices will synchronize the same MAC address for a given IRB:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set vlans vlan100 l3-interface irb.100

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set vlans vlan100 mcae-mac-synchronize

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set vlans vlan200 l3-interface irb.200
```

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set vlans vlan200 mcae-mac-synchronize
```

Let's commit the configuration and examine the changes made to the environment:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# commit and-quit
configuration check succeeds
commit complete
Exiting configuration mode

{master:0}
jedi@AD-1> show vlans

Routing instance        VLAN name             Tag          Interfaces
default-switch          default               1
                                                           et-0/0/35.0*
default-switch          vlan100               100
                                                           ae0.0*
                                                           et-0/0/35.0*
                                                           ge-104/0/0.0*
default-switch          vlan200               200
                                                           et-0/0/35.0*
                                                           ge-104/0/1.0*
```

The output shows that the ae0.0 and ge-104/0/0.0 extended port interfaces are connected to VLAN 100. Additionally, the extended port ge-104/0/1.0 is likewise connected to VLAN 200.

Let's test to see if connectivity is working. First, by testing Host A's connectivity to the irb.100 interface:

```
jedi@Host_A% ping 10.0.1.1
PING 10.0.1.1 (10.0.1.1): 56 data bytes
64 bytes from 10.0.1.1: icmp_seq=0 ttl=64 time=71.979 ms
64 bytes from 10.0.1.1: icmp_seq=1 ttl=64 time=3.675 ms
64 bytes from 10.0.1.1: icmp_seq=2 ttl=64 time=3.693 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=64 time=4.260 ms

--- 10.0.1.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 3.675/20.902/71.979/29.490 ms
```

And now let's make sure intra-VLAN switching is working correctly, by pinging Host B from Host A. In this case, traffic will travel along VLAN 100 via ae0 on satellite 100 and 101, up towards our aggregation devices and then back down towards extended port ge-104/0/0 on satellite 104:

```
jedi@Host_A% ping 10.0.1.3
PING 10.0.1.3 (10.0.1.3): 56 data bytes
64 bytes from 10.0.1.3: icmp_seq=0 ttl=64 time=3.048 ms
64 bytes from 10.0.1.3: icmp_seq=1 ttl=64 time=3.316 ms
64 bytes from 10.0.1.3: icmp_seq=2 ttl=64 time=3.010 ms
64 bytes from 10.0.1.3: icmp_seq=3 ttl=64 time=3.012 ms
```

```
--- 10.0.1.3 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 3.010/3.096/3.316/0.128 ms
```

And lastly, let's verify that inter-VLAN routing is working correctly, by pinging Host C from Host A. In this case, traffic will travel along VLAN 100 via ae0 on satellite 100 and 101, up towards our aggregation devices, where it will then be routed across our IRB interfaces before heading back down via VLAN 200 towards extended port ge-104/0/1 on satellite 104:

```
jedi@Host_A% ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2): 56 data bytes
64 bytes from 10.0.2.2: icmp_seq=0 ttl=63 time=3.783 ms
64 bytes from 10.0.2.2: icmp_seq=1 ttl=63 time=1.543 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=63 time=4.026 ms
64 bytes from 10.0.2.2: icmp_seq=3 ttl=63 time=2.799 ms

--- 10.0.2.2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.543/3.038/4.026/0.978 ms
```

You could also observe this by examining traceroute output:

```
jedi@Host_A% traceroute 10.0.2.2
traceroute to 10.0.2.2 (10.0.2.2), 30 hops max, 40 byte packets
 1  10.0.1.1 (10.0.1.1)  3.692 ms  2.306 ms  2.151 ms
 2  10.0.2.2 (10.0.2.2)  4.419 ms  4.661 ms  5.133 ms
```

Finally, let's examine the output of several commands on the AD-1 to examine the ARP tables and Ethernet switching tables:

```
{master:0}
jedi@AD-1> show ethernet-switching table

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static, C - Control MAC
         SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)


Ethernet switching table : 3 entries, 3 learned
Routing instance : default-switch
    Vlan                MAC                 MAC         Age    Logical                NH        RTR
    name                address             flags              interface              Index     ID
    vlan100             00:50:56:a3:cc:47   DR            -    ge-104/0/0.0           0         0
    vlan100             4c:96:14:40:71:40   DLR           -    ae0.0                  0         0
    vlan200             00:50:56:a3:2c:8e   DLR           -    ge-104/0/1.0           0         0

{master:0}
jedi@AD-1> show arp
MAC Address       Address         Name                    Interface               Flags
00:31:46:e6:04:b8 10.0.0.2        10.0.0.2                et-0/0/35.32769         none
4c:96:14:40:71:40 10.0.1.2        10.0.1.2                irb.100 [ae0.0]         none
00:50:56:a3:cc:47 10.0.1.3        10.0.1.3                irb.100 [ge-104/0/0.0]  none
00:50:56:a3:2c:8e 10.0.2.2        10.0.2.2                irb.200 [ge-104/0/1.0]  none
88:a2:5e:59:9e:15 10.16.0.2       10.16.0.2               et-0/0/0.32769          none
```

```
f4:b5:2f:45:94:b5 10.16.0.6       10.16.0.6              et-0/0/1.32769          none
dc:38:e1:5f:e2:05 10.16.0.10      10.16.0.10             et-0/0/2.32769          none
10:0e:7e:b7:6e:85 10.16.0.14      10.16.0.14             et-0/0/3.32769          none
f4:b5:2f:40:bc:1a 10.16.0.18      10.16.0.18             et-0/0/4.32769          none
fe:00:00:00:00:80 128.0.0.16      fpc0                   bme0.0                  permanent
00:19:e2:51:b7:01 172.16.110.1    172.16.110.1           em0.0                   none
00:31:46:ef:01:3f 192.168.1.1     hypervisor             em2.32768               none
```

The output shows that the VLANs and inter-VLAN routing via IRB interfaces are working as expected. The Ethernet switching table and ARP tables also give an indication of the MAC addresses in use by our hosts, in addition to the ports by which these MAC addresses were learned. This is useful information in the event you need to isolate a given host or better understand where traffic from a given host is being observed.

The resulting configuration is here:

```
groups {
    fusion-config-group {
        interfaces {
            ae0 {
                description “AE Interface to Host A”;
                aggregated-ether-options {
                    lacp {
                        active;
                        periodic fast;
                    }
                }
                unit 0 {
                    family ethernet-switching {
                        vlan {
                            members 100;
                        }
                    }
                }
            }
            irb {
                unit 100 {
                    family inet {
                        address 10.0.1.1/24;
                    }
                }
                unit 200 {
                    family inet {
                        address 10.0.2.1/24;
                    }
                }
            }
            ge-104/0/0 {
                description “GigE to Host B”;
                unit 0 {
                    family ethernet-switching {
                        vlan {
                            members 100;
                        }
                    }
```

```
                }
            }
            ge-104/0/1 {
                description "GigE to Host C";
                unit 0 {
                    family ethernet-switching {
                        vlan {
                            members 200;
                        }
                    }
                }
            }
        }
        vlans {
            vlan100 {
                vlan-id 100;
                l3-interface irb.100;
                mcae-mac-synchronize;
            }
            vlan200 {
                vlan-id 200;
                l3-interface irb.200;
                mcae-mac-synchronize;
            }
        }
    }
}
```

VLAN Autosensing

VLAN autosensing is a Junos Fusion Data Center feature that allows extended ports on satellite devices to dynamically provision VLANs upon observing VLAN-tagged traffic. In the previous examples, where we manually configured VLANs, the extended ports acted as access ports. However, in modern data centers, it is more common to see satellite devices connecting the hosts that run virtualization software on top, such as KVM or VMware. In many of these cases, the extended ports going to hosts will be trunk ports, and a number of different VLANs may be extended. While you could manually configure the extended ports as trunk ports and configure all the expected VLANs on these ports, it's time consuming, and also can consume large amounts of VLAN memory on an aggregation device, especially when these VLANs go unused. At of the time of this writing, Junos Fusion Data Center supports up to 64 satellites and a total of 3000 extended ports. As a result, manually configuring VLANs across all of these interfaces does not scale, and it certainly doesn't help to move the overarching goals of automating the network forward.

With VLAN autosensing, you no longer need to manually enable VLANs on extended ports. You can simply enable these ports as trunk ports, and instead of specifying a set of VLANs, you simply turn on the `vlan-auto-sense` command. As soon as traffic is received on an extended port from a given MAC address with a

VLAN tag associated with it, the satellite will automatically provision the VLAN on that port. The VLAN membership will remain provisioned until there is no more traffic observed for that VLAN on that extended port. This is based on a MAC address age-out timer which is set to 600 seconds by default, but can be modified via the configuration.

NOTE It is important to note that at the time of this book's writing, only a single VLAN tag is supported for VLAN autosensing to work. In other words, double-tagged or Q-in-Q packets can not be sensed automatically via VLAN autosense, and if these packets must be supported, the extended ports must be configured manually.

Let's circle back and change the previously configured VLAN settings on the respective extended port interfaces from manual settings to `vlan-auto-sense`. We also need to change these ports to trunk ports. Our hosts will be modified to ensure that they will be tagging the traffic with 802.1q VLAN tags such that when traffic arrives on our extended ports there is a VLAN tag associated with it:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk

{master:0}[edit groups fusion-config-group]
jedi@AD-1# delete interfaces ae0 unit 0 family ethernet-switching vlan

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ae0 unit 0 family ethernet-switching vlan-auto-sense

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ge-104/0/0 unit 0 family ethernet-switching interface-mode trunk

{master:0}[edit groups fusion-config-group]
jedi@AD-1# delete interfaces ge-104/0/0 unit 0 family ethernet-switching vlan

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ge-104/0/0 unit 0 family ethernet-switching vlan-auto-sense

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ge-104/0/1 unit 0 family ethernet-switching interface-mode trunk

{master:0}[edit groups fusion-config-group]
jedi@AD-1# delete interfaces ge-104/0/1 unit 0 family ethernet-switching vlan

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ge-104/0/1 unit 0 family ethernet-switching vlan-auto-sense
```

Let's commit and then observe the changes that have been made:

```
jedi@AD-1# commit and-quit
configuration check succeeds
commit complete
Exiting configuration mode
```

Let's show the resulting VLAN configuration that has been applied:

```
{master:0}
jedi@AD-1> show vlans

Routing instance        VLAN name             Tag          Interfaces
default-switch          default               1
                                                           et-0/0/35.0*
default-switch          vlan100               100
                                                           et-0/0/35.0*
                                                           ge-104/0/0.0*
default-switch          vlan200               200
                                                           et-0/0/35.0*
                                                           ge-104/0/1.0*
```

And AD-1 has learned of the existence of VLAN 100 on extended port ge-104/0/0.0 towards Host_B and also VLAN 200 on extended port ge-104/0/1/0 towards Host_C. This is due to periodic traffic that is normally being transmitted from these hosts. Noticeably absent from this list, however, is our ae0 extended port towards Host A. Let's initiate some traffic from Host A and see if anything changes:

```
jedi@Host_A% ping 10.0.1.1
PING 10.0.1.1 (10.0.1.1): 56 data bytes
64 bytes from 10.0.1.1: icmp_seq=0 ttl=64 time=2.563 ms
64 bytes from 10.0.1.1: icmp_seq=1 ttl=64 time=3.894 ms
64 bytes from 10.0.1.1: icmp_seq=2 ttl=64 time=3.258 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=64 time=4.039 ms

--- 10.0.1.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.563/3.438/4.039/0.585 ms
```

Now, let's take another look at our VLANs and see if this traffic has triggered VLAN autosensing:

```
{master:0}
jedi@AD-1> show vlans

Routing instance        VLAN name             Tag          Interfaces
default-switch          default               1
                                                           et-0/0/35.0*
default-switch          vlan100               100
                                                           ae0.0*
                                                           et-0/0/35.0*
                                                           ge-104/0/0.0*
default-switch          vlan200               200
                                                           et-0/0/35.0*
                                                           ge-104/0/1.0*
```

You can see the addition of the ae0.0 interface to VLAN 100, indicating that it was learned dynamically upon the receipt of VLAN tagged traffic upon this interface.

You can also get additional information via the `show ethernet-switching interfaces satellite` output indicating which VLANs are associated with which extended ports, whether these ports are configured as access ports or trunk ports, and whether VLAN autosense has been enabled:

```
{master:0}
jedi@AD-1> show ethernet-switching interface satellite
Interface       Instance        PORT-MODE       Auto-Sense      VLAN
ae0.0           default-switch  TRUNK           ENABLE
                                                                vlan100
ge-104/0/0.0    default-switch  TRUNK           ENABLE
                                                                vlan100
ge-104/0/1.0    default-switch  TRUNK           ENABLE
                                                                vlan200
```

A third command, `show vlans satellite` shows us similar information regarding VLAN associated on the extended ports:

```
{master:0}
jedi@AD-1> show vlans satellite
Instance        VLAN name           Tag         Interfaces
default-switch  default             1
                                                et-0/0/35.0
default-switch  vlan100             100
                                                ae0.0 (AS)
                                                et-0/0/35.0
                                                ge-104/0/0.0 (AS)
default-switch  vlan200             200
                                                et-0/0/35.0
                                                ge-104/0/1.0 (AS)
```

VLAN autosensing is a powerful feature built into Junos Fusion Data Center. When you look at modern data centers with their thousands of ports managed via a single-pane-of-glass, automation is the cornerstone of any solution which is designed to scale. VLAN autosense not only saves time and eliminates mistakes that can be made when making manual configuration changes, but it also helps to save valuable VLAN table memory on our aggregation devices, by only enabling VLANs on extended ports when necessary, and decommissioning them when no longer in use.

This concludes the section on VLAN autosense, but before leaving let's re-examine the configuration that was used to enable this feature:

```
groups {
    fusion-config-group {
        interfaces {
            ae0 {
                description “AE Interface to Host A”;
                aggregated-ether-options {
                    lacp {
                        active;
                        periodic fast;
```

```
                }
            }
            unit 0 {
                family ethernet-switching {
                    interface-mode trunk;
                    vlan-auto-sense;
                }
            }
        }
        ge-104/0/0 {
            description "GigE to Host B";
            unit 0 {
                family ethernet-switching {
                    interface-mode trunk;
                    vlan-auto-sense;
                }
            }
        }
        ge-104/0/1 {
            description "GigE to Host C";
            unit 0 {
                family ethernet-switching {
                    interface-mode trunk;
                    vlan-auto-sense;
                }
            }
        }
    }
  }
}
```

Local Switching

In the default Junos Fusion topology, Layer 2 traffic from one host on a satellite device to another host on the same satellite device is forwarded to the aggregation device. In other words, the satellite device simply forwards any received traffic upstream towards the aggregation device without any additional processing.

This obviously does not represent the best case scenario as it results in a ping-pong of traffic that could better be served locally, as opposed to unnecessarily using the bandwidth on the uplink interfaces. Due to this hair-pinning of traffic, the uplinks can become over utilized and can also result in higher latencies between hosts.

In order to rectify this issue, you can configure local switching on the various satellite devices such that Layer 2 unicast traffic is switched locally on a satellite so long as the destination MAC address is reachable on the same satellite.

Let's take a look at this by adding a fourth host to our topology, Host D, as can be seen in Figure 5.6.

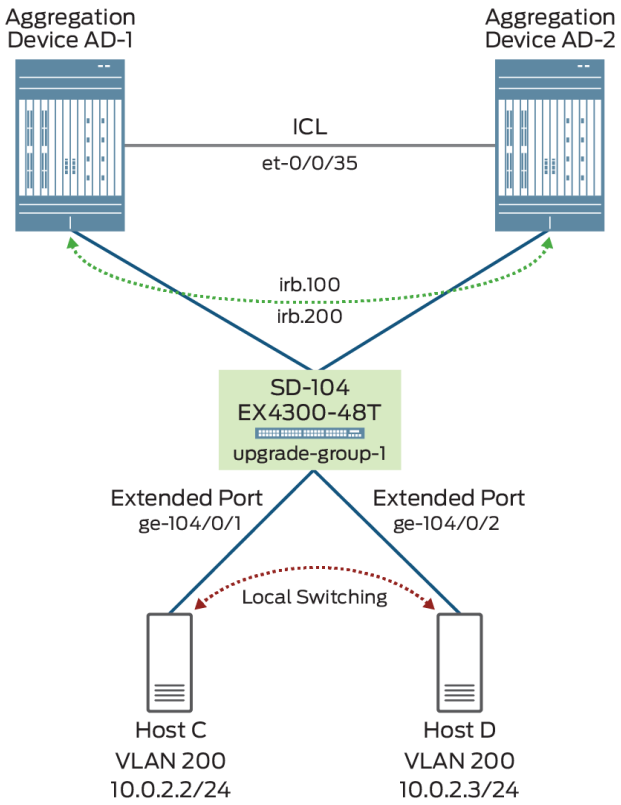

Figure 5.6 Local Switching

This new host will belong to the same VLAN as Host C and as they are both tethered to the same satellite, they are perfect candidates for local switching. Let's add the new host on extended port ge-104/0/2 and enable `local-switching` for FPC 104. Notice it's under the [groups fusion-config-group] stanza in order to ensure the configuration is propagated to the other aggregation device, AD-2:

```
{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ge-104/0/2 description "GigE to Host D"

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ge-104/0/2 unit 0 family ethernet-switching interface-mode trunk

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set interfaces ge-104/0/2 unit 0 family ethernet-switching vlan-auto-sense

{master:0}[edit groups fusion-config-group]
jedi@AD-1# set forwarding-options satellite fpc 104 local-switching
```

Unfortunately, there is no way to demonstrate that this is actually working via any available show commands, so you will just have to trust that everything is working as expected and that any non-BUM traffic is getting switched locally.

The configuration is here:

```
groups {
    fusion-config-group {
        interfaces {
            ge-104/0/1 {
                description "GigE to Host C";
                unit 0 {
                    family ethernet-switching {
                        interface-mode trunk;
                        vlan-auto-sense;
                    }
                }
            }
            ge-104/0/2 {
                description "GigE to Host D";
                unit 0 {
                    family ethernet-switching {
                        interface-mode trunk;
                        vlan-auto-sense;
                    }
                }
            }
        }
        forwarding-options {
            satellite {
                fpc 104 {
                    local-switching;
                }
            }
        }
    }
}
```

Conclusion: Data Center Simplicity

If you've made it this far, I offer abundant applause. I truly hope you were able to learn something new and hopefully I've been able to impart some of my excitement for this fascinating new technology. Juniper has provided their customer base with a wide variety of different fabric offerings, but if you've followed this *Day One* book, I truly believe you've learned one of the best new data center technologies that Juniper has engineered.

Junos Fusion Data Center marks a significant leap forward for Juniper in the evolving world of Ethernet fabric architectures. It combines all the benefits of Virtual Chassis Fabric, such as single-pane-of-glass management, while at the same time yielding scaling benefits that put it on par with a more traditional IP fabric. But compared to an IP fabric, you should now clearly see that Junos Fusion Data Center is a much easier technology to deploy. With such benefits as Software Upgrade Groups, management of satellites is a snap, and features such as VLAN autosensing remove the cumbersome need to manually configure Layer 2 elements throughout your environment. It seems that Juniper has created Junos Fusion first and foremost with a focus on simplicity.

By now, you've learned the basics of a Junos Fusion architecture, and you should understand it's underpinnings and all the underlying machinery from a technical standpoint. You've learned how to configure a basic topology utilizing a single aggregation device before delving into the more robust, resilient, and preferred architecture involving multiple aggregation devices. You've learned how to successfully verify, ascertain, and operate a working Junos Fusion architecture once deployed, and finally, you've learned how to connect to hosts, which is after all, the main point of deploying a Junos Fusion architecture in the first place.

But this is not the end of the story. This is, in fact, just the beginning. To truly master the technology, I encourage you to peruse Juniper's technical documentation, as this book only covers the basic building blocks to get a Junos Fusion Data Center up and running. There are many more advanced features that simply couldn't be covered. Furthermore, as this is an evolving technology, more enhancements will be introduced in the near future – everything from increased capabilities, improved scalability, features that aid in simplicity, and integration with emerging technologies such as EVPN. There is much, much more to come.

It has been a pleasure writing this book and I will continue to remain involved in educating the community with forthcoming new developments in Junos Fusion Data Center as they are released. To that end, I would encourage readers to visit my blog at http://www.shortestpathfirst.net. Not only will I occasionally write articles covering this technology but I plan on producing several videos which will demonstrate the configuration and day-to-day operation of a Junos Fusion Data Center. There were many features I did not cover in this book – things such as Junos Space Network Director integration with Junos Fusion, loop detection mechanisms, and multicast replication. So please, check in regularly.

Stefan Fouant, December 2017

Appendix: Configurations

AD-1 Configuration

```
groups {
    fusion-config-group {
        when {
            peers [ 172.16.110.24 172.16.110.25 ];
        }
        chassis {
            aggregated-devices {
                ethernet {
                    device-count 100;
                }
            }
            satellite-management {
                fpc 100 {
                    alias Rack1-QFX5100-A;
                    cascade-ports et-0/0/0;
                    uplink-failure-detection {
                        candidate-uplink-policy 2-link-policy;
                    }
                }
                fpc 101 {
                    alias Rack1-QFX5100-B;
                    cascade-ports et-0/0/1;
                }
                fpc 102 {
                    alias Rack2-QFX5100-A;
                    cascade-ports et-0/0/2;
                }
                fpc 103 {
                    alias Rack2-QFX5100-B;
                    cascade-ports et-0/0/3;
```

```
        }
        fpc 104 {
            alias Rack3-EX4300-A;
            cascade-ports et-0/0/4;
        }
        uplink-failure-detection;
        upgrade-groups {
            upgrade-group-1 {
                satellite [ 100 102 104 ];
            }
            upgrade-group-2 {
                satellite [ 101 103 ];
            }
        }
        auto-satellite-conversion {
            satellite [ 102 103 104 ];
        }
    }
}
interfaces {
    et-0/0/0 {
        cascade-port;
    }
    et-0/0/1 {
        cascade-port;
    }
    et-0/0/2 {
        cascade-port;
    }
    et-0/0/3 {
        cascade-port;
    }
    et-0/0/4 {
        cascade-port;
    }
    ae0 {
        description “AE Interface to Host A”;
        aggregated-ether-options {
            lacp {
                active;
                periodic fast;
            }
        }
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan-auto-sense;
            }
        }
    }
    ge-100/0/0 {
        ether-options {
            802.3ad ae0;
        }
    }
    ge-101/0/0 {
        ether-options {
            802.3ad ae0;
```

```
        }
    }
    irb {
        unit 100 {
            family inet {
                address 10.0.1.1/24;
            }
        }
        unit 200 {
            family inet {
                address 10.0.2.1/24;
            }
        }
    }
    ge-104/0/0 {
        description “GigE to Host B”;
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan-auto-sense;
            }
        }
    }
    ge-104/0/1 {
        description “GigE to Host C”;
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan-auto-sense;
            }
        }
    }
    ge-104/0/2 {
        description “GigE to Host D”;
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan-auto-sense;
            }
        }
    }
}
forwarding-options {
    satellite {
        fpc 104 {
            local-switching;
        }
    }
}
policy-options {
    satellite-policies {
        candidate-uplink-port-policy 2-link-policy {
            minimum-links 2;
        }
    }
}
vlans {
    vlan100 {
```

```
                vlan-id 100;
                l3-interface irb.100;
                mcae-mac-synchronize;
            }
            vlan200 {
                vlan-id 200;
                l3-interface irb.200;
                mcae-mac-synchronize;
            }
        }
    }
}
apply-groups fusion-config-group;
system {
    host-name AD-1;
    root-authentication {
    }
    login {
        user ftp {
            uid 2003;
            class read-only;
        }
        user sync_user {
            uid 2001;
            class super-user;
            authentication {
            }
        }
    }
    services {
        ssh;
        netconf {
            ssh;
        }
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
    commit {
        peers-synchronize;
        peers {
            172.16.110.25 {
                user sync_user;
            }
        }
    }
    processes {
        dhcp-service {
            traceoptions {
```

```
                file dhcp_logfile size 10m;
                level all;
                flag all;
            }
        }
    }
}
chassis {
    satellite-management {
        redundancy-groups {
            chassis-id 1;
            dc-pod-1 {
                redundancy-group-id 1;
                peer-chassis-id 2 {
                    inter-chassis-link et-0/0/35;
                    liveness-detection {
                        minimum-interval 2000;
                        multiplier 3;
                        transmit-interval {
                            minimum-interval 2000;
                        }
                    }
                }
                satellite all;
            }
        }
    }
}
interfaces {
    et-0/0/35 {
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan {
                    members all;
                }
            }
        }
    }
    em0 {
        unit 0 {
            family inet {
                address 172.16.110.24/24;
            }
        }
    }
    irb {
        unit 0;
    }
}
forwarding-options {
    storm-control-profiles default {
        all;
    }
}
routing-options {
    static {
        route 0.0.0.0/0 next-hop 172.16.110.1;
```

```
    }
}
protocols {
    lldp {
        interface all;
    }
    lldp-med {
        interface all;
    }
    igmp-snooping {
        vlan default;
    }
}
vlans {
    default {
        vlan-id 1;
        l3-interface irb.0;
    }
}
```

AD-2 Configuration

```
groups {
    fusion-config-group {
        when {
            peers [ 172.16.110.24 172.16.110.25 ];
        }
        chassis {
            aggregated-devices {
                ethernet {
                    device-count 100;
                }
            }
            satellite-management {
                fpc 100 {
                    alias Rack1-QFX5100-A;
                    cascade-ports et-0/0/0;
                    uplink-failure-detection {
                        candidate-uplink-policy 2-link-policy;
                    }
                }
                fpc 101 {
                    alias Rack1-QFX5100-B;
                    cascade-ports et-0/0/1;
                }
                fpc 102 {
                    alias Rack2-QFX5100-A;
                    cascade-ports et-0/0/2;
                }
                fpc 103 {
                    alias Rack2-QFX5100-B;
                    cascade-ports et-0/0/3;
                }
                fpc 104 {
                    alias Rack3-EX4300-A;
                    cascade-ports et-0/0/4;
```

```
        }
        uplink-failure-detection;
        upgrade-groups {
            upgrade-group-1 {
                satellite [ 100 102 104 ];
            }
            upgrade-group-2 {
                satellite [ 101 103 ];
            }
        }
        auto-satellite-conversion {
            satellite [ 102 103 104 ];
        }
    }
}
interfaces {
    et-0/0/0 {
        cascade-port;
    }
    et-0/0/1 {
        cascade-port;
    }
    et-0/0/2 {
        cascade-port;
    }
    et-0/0/3 {
        cascade-port;
    }
    et-0/0/4 {
        cascade-port;
    }
    ae0 {
        description “AE Interface to Host A”;
        aggregated-ether-options {
            lacp {
                active;
                periodic fast;
            }
        }
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan-auto-sense;
            }
        }
    }
    ge-100/0/0 {
        ether-options {
            802.3ad ae0;
        }
    }
    ge-101/0/0 {
        ether-options {
            802.3ad ae0;
        }
    }
    irb {
        unit 100 {
```

```
            family inet {
                address 10.0.1.1/24;
            }
        }
        unit 200 {
            family inet {
                address 10.0.2.1/24;
            }
        }
    }
    ge-104/0/0 {
        description “GigE to Host B”;
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan-auto-sense;
            }
        }
    }
    ge-104/0/1 {
        description “GigE to Host C”;
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan-auto-sense;
            }
        }
    }
    ge-104/0/2 {
        description “GigE to Host D”;
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan-auto-sense;
            }
        }
    }
}
forwarding-options {
    satellite {
        fpc 104 {
            local-switching;
        }
    }
}
policy-options {
    satellite-policies {
        candidate-uplink-port-policy 2-link-policy {
            minimum-links 2;
        }
    }
}
vlans {
    vlan100 {
        vlan-id 100;
        l3-interface irb.100;
        mcae-mac-synchronize;
    }
```

```
            vlan200 {
                vlan-id 200;
                l3-interface irb.200;
                mcae-mac-synchronize;
            }
        }
    }
}
apply-groups fusion-config-group;
system {
    host-name AD-2;
    root-authentication {
    }
    login {
        user ftp {
            uid 2002;
            class read-only;
        }
        user sync_user {
            uid 2001;
            class super-user;
            authentication {
            }
        }
    }
    services {
        ssh;
        netconf {
            ssh;
        }
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
    extensions {
        providers {
            juniper {
                license-type juniper deployment-scope commercial;
            }
            chef {
                license-type juniper deployment-scope commercial;
            }
        }
    }
    commit {
        peers-synchronize;
        peers {
            172.16.110.24 {
                user sync_user;
```

```
            }
        }
    }
    processes {
        app-engine-virtual-machine-management-service {
            traceoptions {
                level notice;
                flag all;
            }
        }
    }
}
chassis {
    satellite-management {
        redundancy-groups {
            chassis-id 2;
            dc-pod-1 {
                redundancy-group-id 1;
                peer-chassis-id 1 {
                    inter-chassis-link et-0/0/35;
                    liveness-detection {
                        minimum-interval 2000;
                        multiplier 3;
                        transmit-interval {
                            minimum-interval 2000;
                        }
                    }
                }
                satellite all;
            }
        }
    }
}
interfaces {
    et-0/0/35 {
        unit 0 {
            family ethernet-switching {
                interface-mode trunk;
                vlan {
                    members all;
                }
            }
        }
    }
    em0 {
        unit 0 {
            family inet {
                address 172.16.110.25/24;
            }
        }
    }
}
forwarding-options {
    storm-control-profiles default {
        all;
    }
}
routing-options {
```

```
    static {
        route 0.0.0.0/0 next-hop 172.16.110.1;
    }
}
protocols {
    lldp {
        interface all;
    }
    lldp-med {
        interface all;
    }
    igmp-snooping {
        vlan default;
    }
    rstp {
        interface all;
    }
}
```

Additional Resources

Juniper Resources

The *J-Net Community Forum* is dedicated to sharing information, best practices, and questions about Juniper products, technologies, and solutions. Register to participate for this no-cost forum and get free access to the *Day One* library.
http://forums.juniper.net/jnet

Junos OS Technical documentation includes everything you need to understand and configure all aspects of Junos Fusion Data Center.
https://www.juniper.net/techpubs/en_US/release-independent/junos/information-products/pathway-pages/junos-fusion/junos-fusion-data-center/product/index.html

The *Junos Fusion Hardware and Software Compatibility Matrix* in the Juniper Software Center provides a way to access all Junos OS and satellite software for any Junos Fusion Data Center installation, and also provides a matrix of the Junos OS and satellite software requirements for any Junos Fusion Data Center setup.
https://www.juniper.net/support/downloads/solutions/fusion/

The *Juniper Data Center Network Design and Architecture Center* contains a wealth of learning information: articles, videos, solutions, guides, and more.
https://www.juniper.net/documentation/en_US/design-and-architecture/data-center/index.html

The *Cloud Enabled Enterprise Data Center* portal contains a wealth of information around all aspects of Juniper's Data Center strategy, including case studies, implementation guides, solutions briefs and white papers.
https://www.juniper.net/us/en/solutions/cloud-enabled-enterprise/data-center/

Resources from author, Stefan Fouant

Stefan Fouant's blog covering everything from network architecture and design, data center design and infrastructure virtualization, DDoS mitigation and detection, disaster recovery planning, IPv6 planning and deployment, multicast, MPLS and BGP/MPLS/VxLAN-based VPNs, SDN, anything having to do with Juniper certifications, general technology trends, and overall information security best practices.
http://www.shortestpathfirst.net/