HP IUM Fundamentals

IUM Processes & Components. Encapsulators

IUM Processes - CORBA

- IUM Uses a 3rd Party CORBA implementation called Visibroker from Borland (formerly Visigenics).

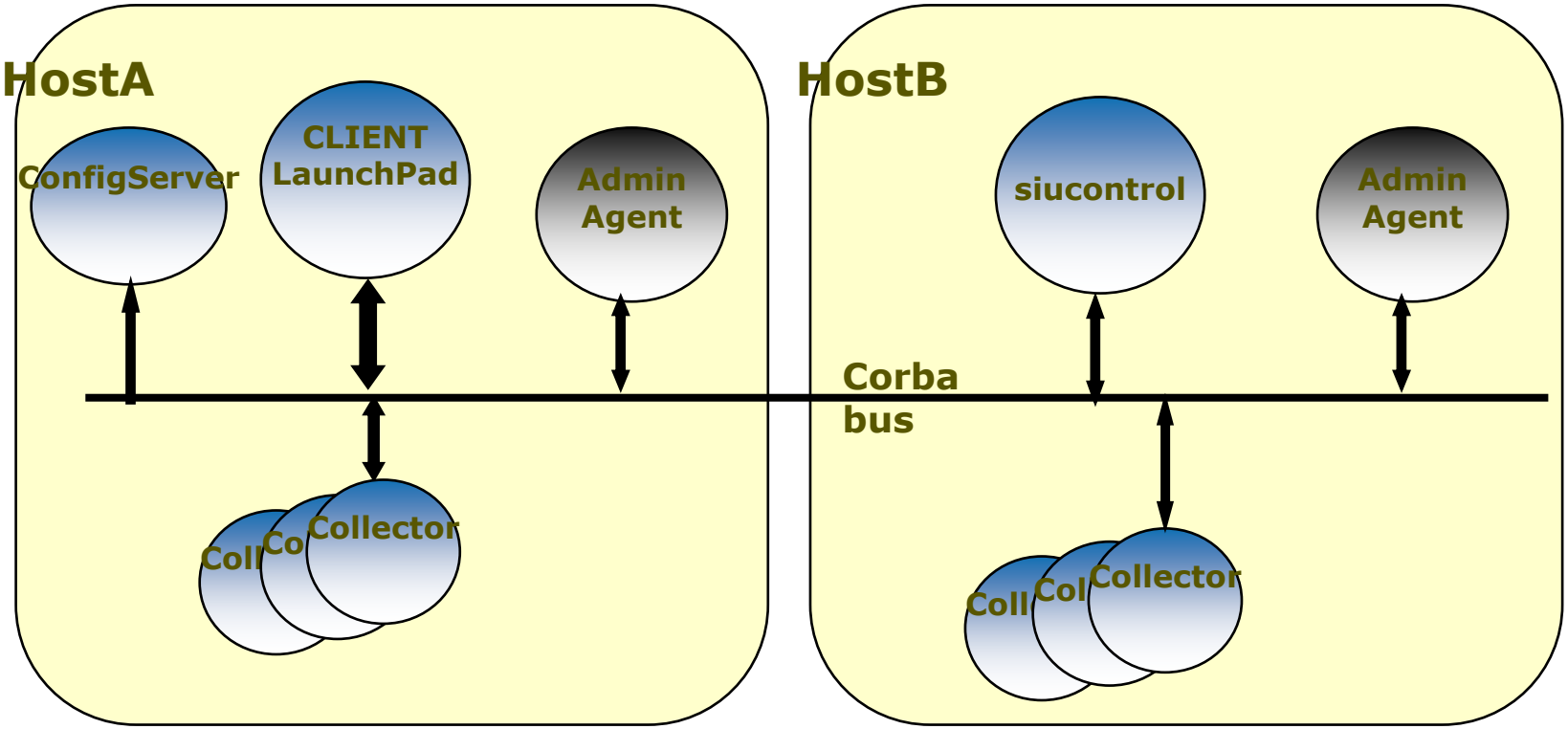
- IUM uses the Visibroker runtime libraries to provide a transport layer for all communications within IUM

- The Visibroker libraries are installed automatically as part of the IUM installation process

- The runtime libraries are installed in C:\SIU\lib\vbapp.jar

IUM Processes - CORBA

- The CORBA Bus or Transport Layer allows all communication between components in IUM

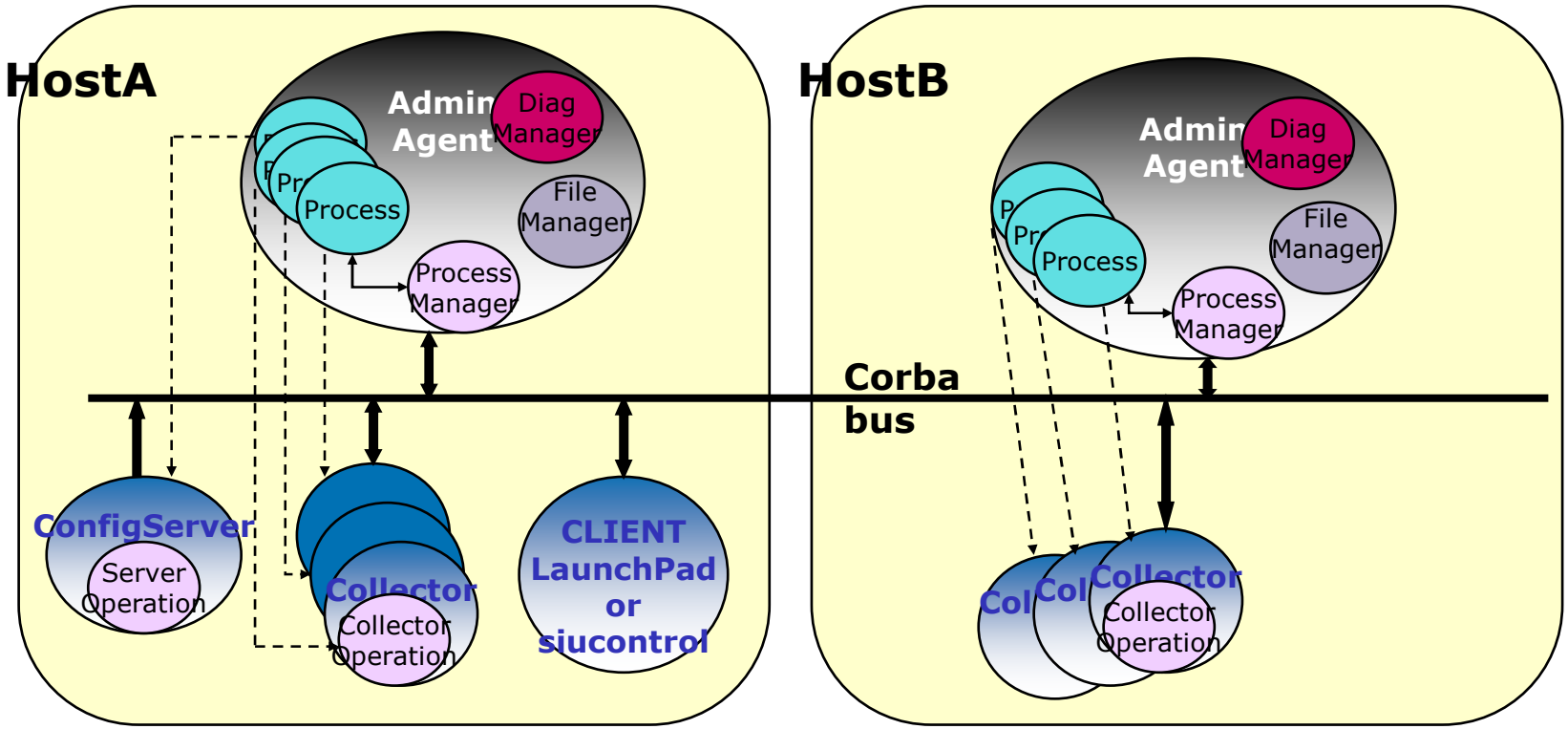


IUM Processes - CORBA

- Communication is managed by each process in IUM (including collectors) has its own file called an IOR. IOR = Interoperable Object Reference.
- The IOR contains the IP address and port number used by that process.
- By exchanging the IOR between the Config Server and each collector, the Config Server can locate each collector and manage it.
- IORs are stored in C:\SIU\var\ConfigServer for the Config Server and C:\SIU\var\<collector_name> for each collector.

IUM Processes – Admin Agent

- Admin Agent manages all IUM processes remotely



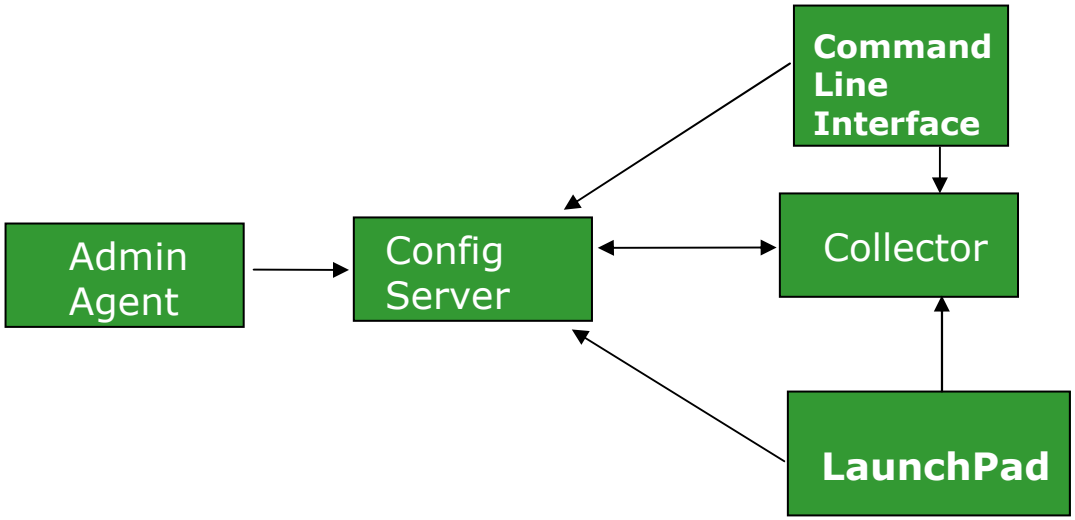
IUM Processes – Admin Agent

- The AdminAgent:
  - Requires **SIU.ini** to start.
  - Requires **SIUJava.ini** to start if AdminAgentServer is starting for the first time and it will be launching the ConfigServer as one of its child processes.
  - Reads HOSTID property from **SIU.ini** for self identification, or defaults to system's host name.
  - Reads SIUJAVAINI property from **SIU.ini** for pointer to **SIUJava.ini**.
  - AdminAgentServer starts the ConfigServer process (STARTCONFIGSERVER=true in **SIU.ini**).

- The installation process installs the Admin Agent as a process on each IUM host machine – NT Service or Unix daemon.

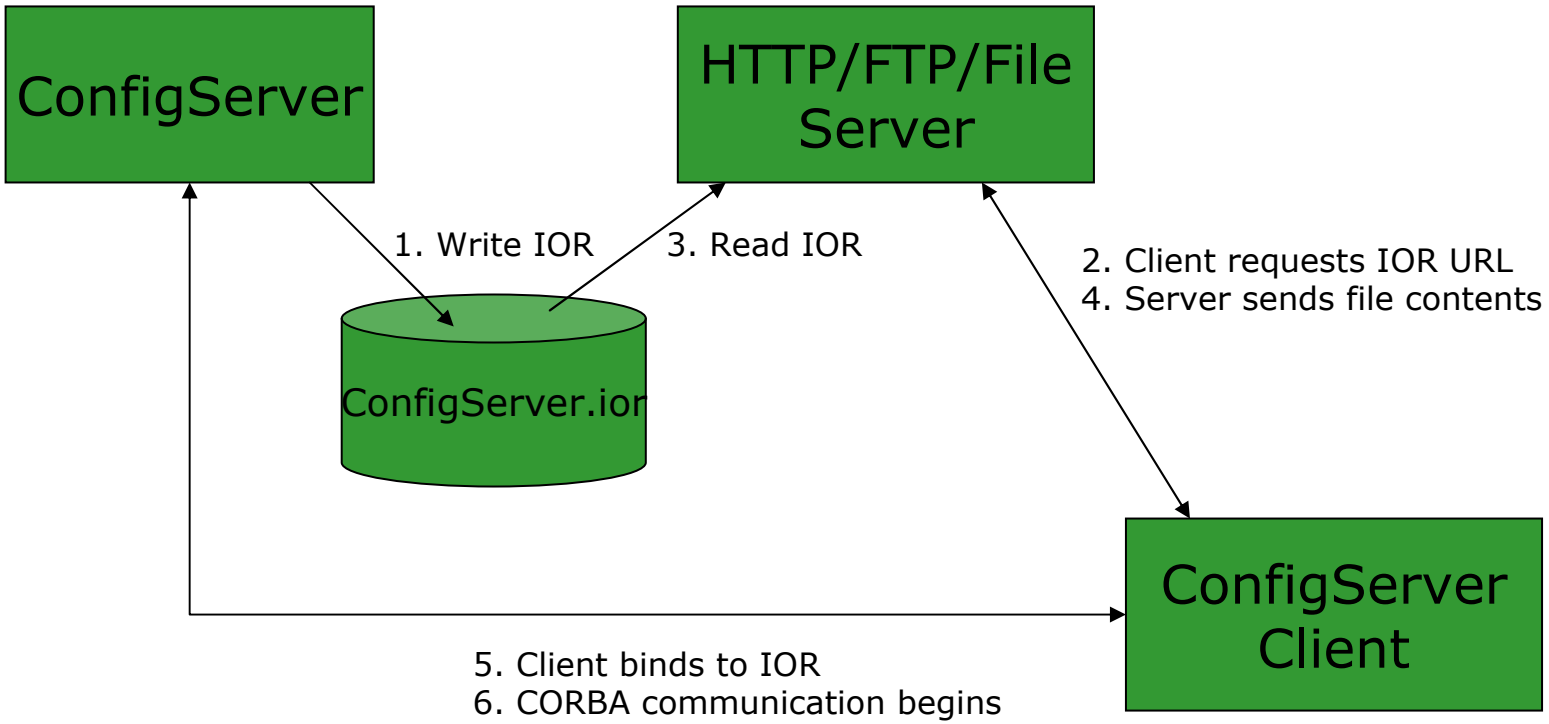
IUM Processes – Config Server

- The Config Server stores and manages all configuration information for a IUM deployment.
- The Config Server is installed on one host.
- The Config Server is run as a process controlled by the host Admin Agent.
- At installation the Config Server writes its CORBA address to a file (**ConfigServer.ior**) and populates the IOR at startup.



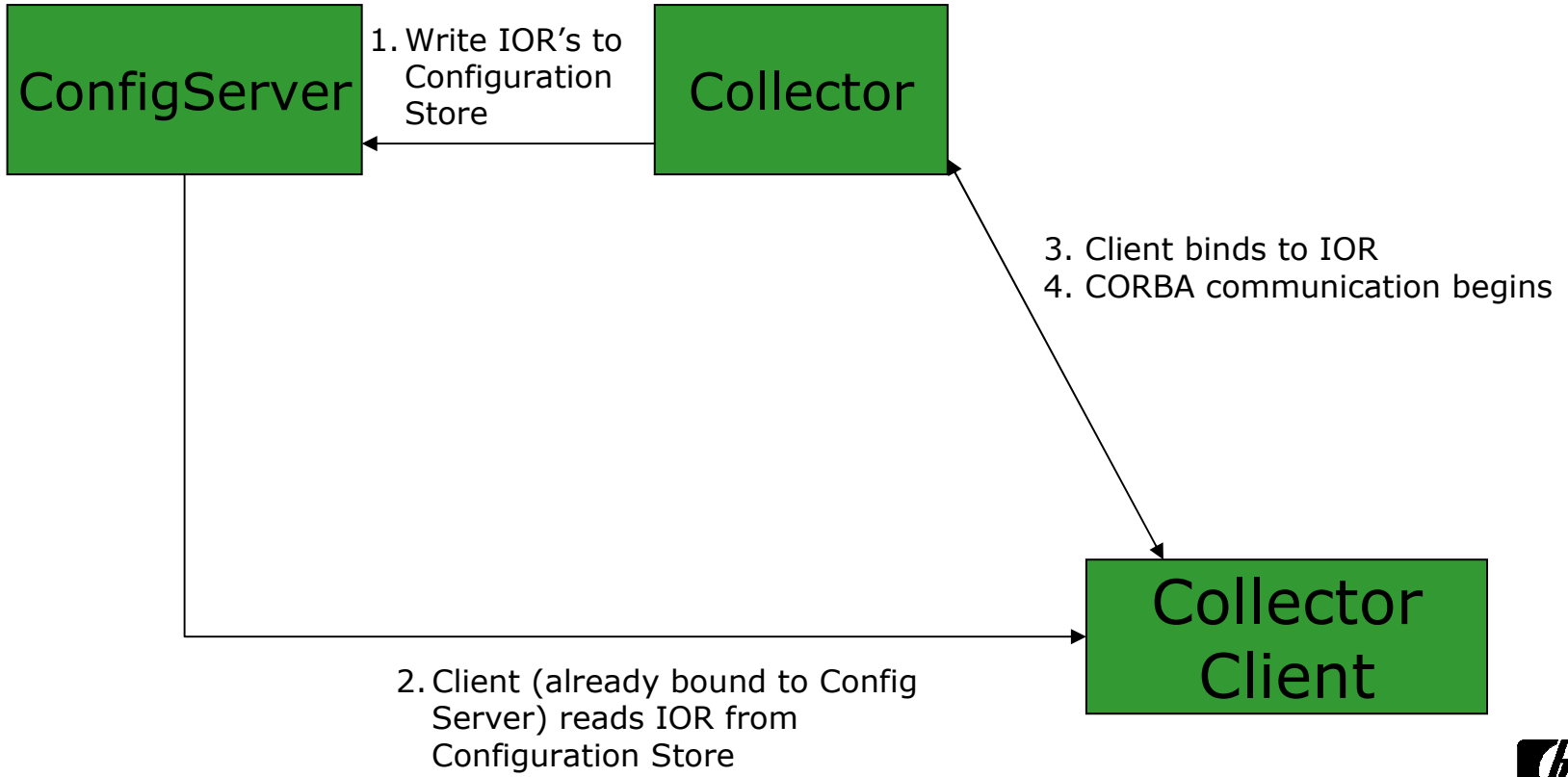
IUM Processes – Config Server

- All hosts and collectors download their configurations from the Config Server when they start up. They use a URL to locate the IOR on the Config Server host machine.
- The Config Server runs either httpd or ftpd on its host machine, so that other hosts and collectors can access the Config Server IOR.



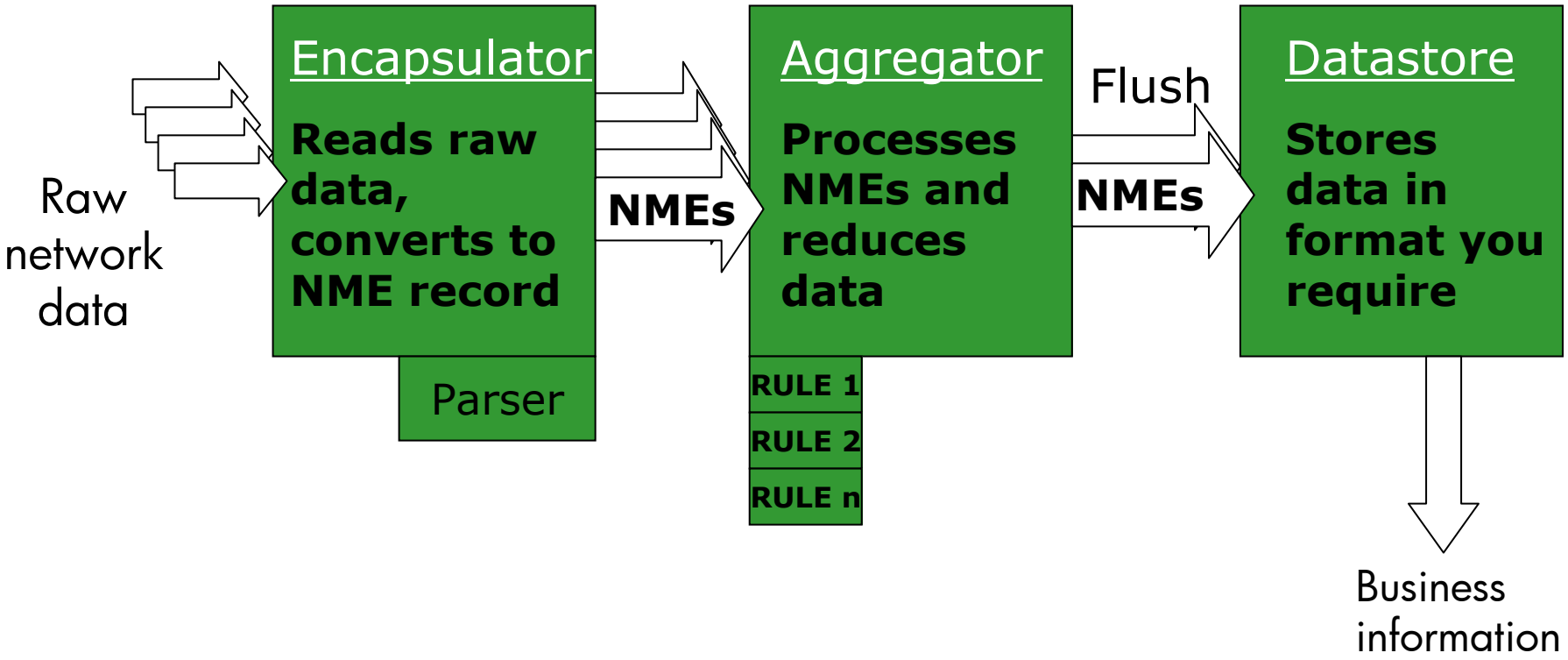
IUM Processes – Config Server

- Collectors also have IORs.
- Collectors register their IORs with the Config Server each time they start up.



IUM Components – Collectors

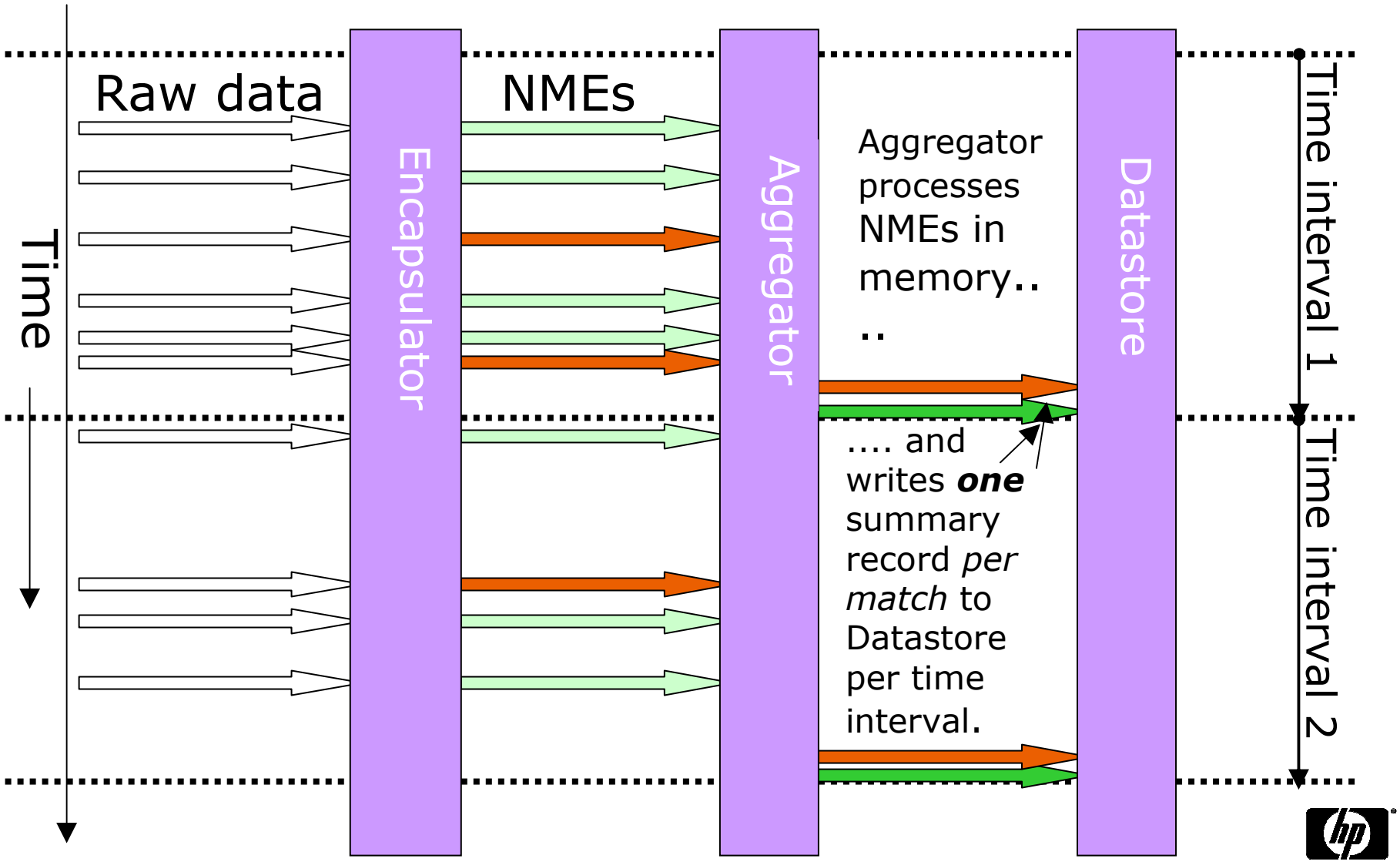
- Which are the components inside a Collector?



IUM Components – Collectors

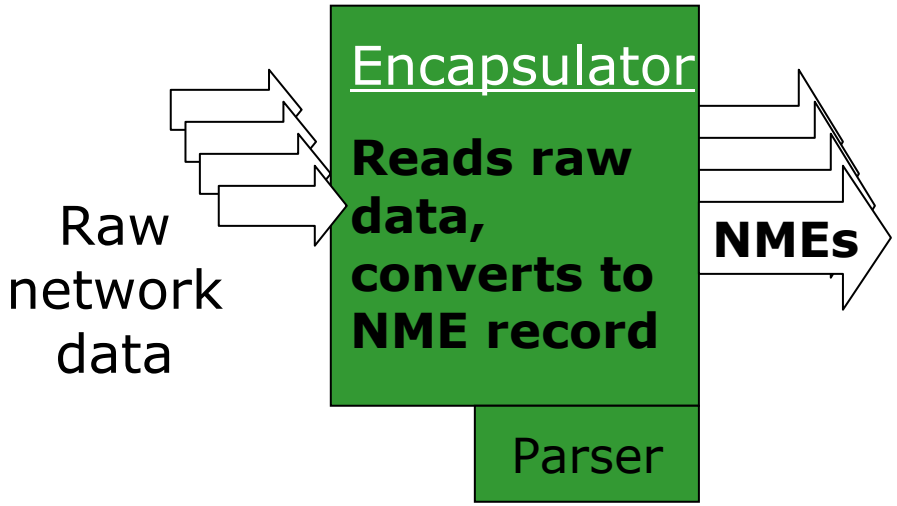
- Data Workflow
  - IUM traditionally uses a time-based data flow. Data flows through a Collector based on time intervals. For example, a router timestamp provides a time interval for usage data.
  - In the IUM GPRS option, data is processed using a data-set model. Data is grouped into a logical set and is passed through the collector and stored as a dataset.

IUM Components – Collectors



IUM Components – Encapsulator

- **The Encapsulator**:
  - Reads data from a network data source
  - Converts input data into an internal record format called an NME
  - Determines when the data (NMEs) is stored in the datastore.
- You may need to configure or change the parser depending on the input structure of the data, e.g. field and record delimiters



IUM Components – Encapsulator

- Some of the major components of an Encapsulator are:
  - **Parser**
    - Supports ASCII (delimiter parser) and Binary (Offset Parser) formats
  - **Data Sources**
    - File based
    - Network based (UDP, telnet, SNMP, et al)
  - **File Manipulation**
    - File Rolling (Mmddyyyy policy)
    - Control File

IUM Components – Encapsulator

There are several different Encapsulator types in IUM:

CollectorEncapsulator → Reads usage data from another collector

IPAcctgEncapsulator → Reads IP accounting data from Cisco Routers

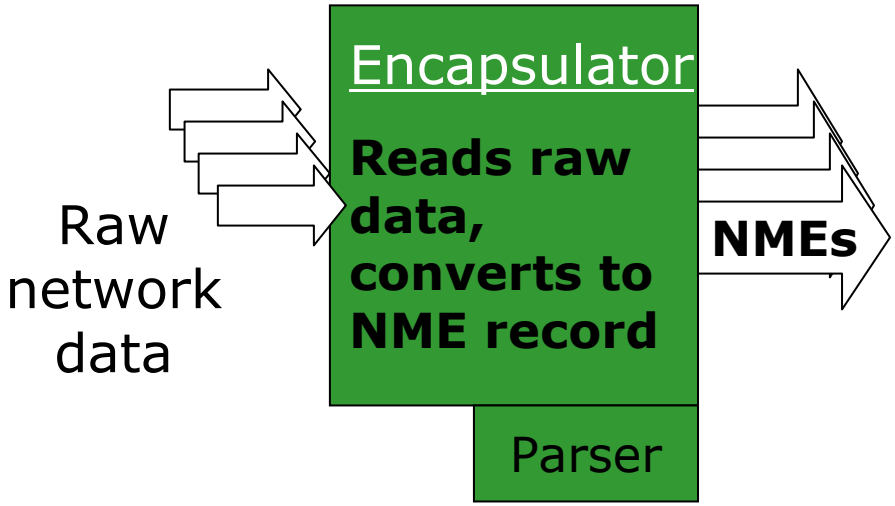
PollingMuxEncapsulator → Co-ordinates the activity of several separate encapsulators that poll

RecordEncapsulator → Processes event records from a file-based or record-based data source

SNMPEncapsulator → Polls instances of SNMP MIB variables from a target SNMP agent

IUM Components – Encapsulator

- The type of the Encapsulator must be chosen based on the input source
- You may need to configure or change the "Parser" depending on the input structure, e.g. Field and Record delimiters



IUM Components – Encapsulator

- The Parser is part of the **Encapsulator**. They translate raw incoming data records into fields that can be placed in an NME. For example, fixed IP input log.

- A Parser is required whenever the data source is not another collector – when the data source is a **network device or file**.

- To Edit a Parser, use the LaunchPad -> Customize Collector. The Attributes are the fields the parser breaks the input record into. You can modify existing parsers to suit your needs.

IUM Components – Encapsulator

There exist a number of pre-configured Parsers:

1. DelimiterParser
2. GroupDelimiterParser
3. NameValueParser
4. NetflowParser
5. NetflowProtocolParser
6. OffsetParser
7. PerlRegexParser
8. ProxyParser
9. RadiusParser
10. RegexParser

…

Select a Parser based on the type of input data.

IUM Components – Encapsulator

- Encapsulators transform input data into NMEs (Normalized Metered Events)
- Metered Events can be:
  - Session Events (start, stop, …)
  - Usage Events (bytes, seconds, …)
  - Resource Events (disk space, …)
- An NME is a record of a Metered Event in a format that can be decode and processed by IUM components

SrcIP	DstIP	NextHop	Numbytes	Start Time	End Time	SrcPort	DstPort

IUM Components – Encapsulator

- NMEs are composed of fields or **Attributes** that correspond with the various fields of some usage event
- The attributes in an NME can be of different types, depending on the type of data being processed. (For example, String, Integer, IP Address, Time
- There is a list of pre-configured NME Attribute names and their types. (For example, StartTime, SrcIP,NumBytes)
- The NME Schema is the sum of all available NME types
- The Encapsulator and Parser populate the NME with data that corresponds to the fields (Attributes) defined
- The NME Schema is a common map shared by all NMEs which describes each field in the NME

IUM Components – Encapsulator

- Structured NMEs are a substantial improvement over traditional flat NMEs.
- Structured NMEs store complete information about the hierarchical arrangement of complex data records
- Structured NMEs contain information about which data fields have been set and which have not.
- Structured NMEs can handle optional data fields.
- Structured NMEs retain all the hierarchical information of structured data

IUM Components – Encapsulator

- A Structured NME with an Array of Sub-NMEs

CreateTime	NumRecord	FileName	Data
Time	*int*	*string*	

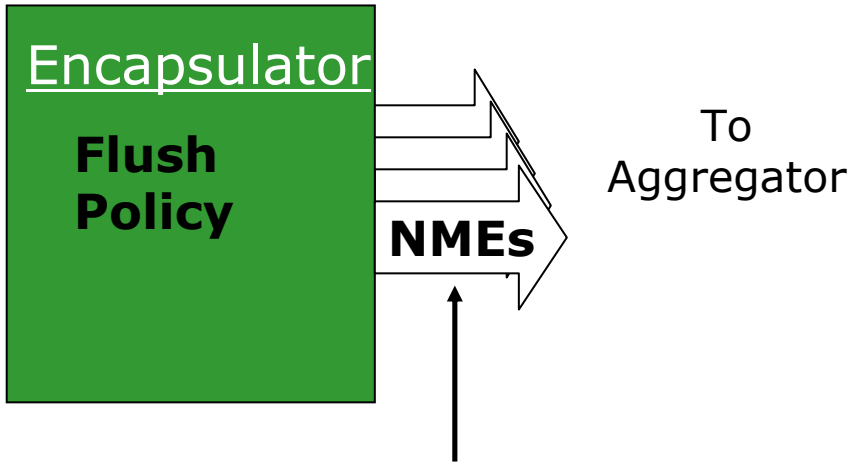
	StartTime	EndTime	SrcIP	DstIP	SrcPort	DstPort	NumBytes	NumPackets
Data[0]:	*Time*	*Time*	*IPAddress*	*IPAddress*	*Port*	*Port*	*long*	*int*
Data[1]:	*Time*	*Time*	*IPAddress*	*IPAddress*	*Port*	*Port*	*long*	*int*
Data[2]:	*Time*	*Time*	*IPAddress*	*IPAddress*	*Port*	*Port*	*long*	*int*

IUM Components – Encapsulator

- ***EndTime*** is an ***NME Attribute***
- Every input source must have EndTime
- End Time is configured in the Parser and used to
  - determine filenames
  - determine flush times
  - for recovery information
  - for history information in support of queries
- Session Sources must also have a *StartTime*
  - to determine duration of active vs. inactive sessions

IUM Components – Encapsulator

- The Flush Policy is part of the Encapsulator
- It is a Time Interval which controls when NMEs are moved from Aggregation to the Datastore.
- Some Encapsulators do not need a Flush Policy, they use a Query Interval.



Flush Policy examines NMEs as they are passed to the Aggregator and determines when the flush is necessary.

IUM Components – Encapsulator

- Some Encapsulators do not need a Flush Policy. For example, Collector Encapsulator, SNMP Encapsulator and Reporting Encapsulator.
- These type of Encapsulators use a Query Interval instead.
- The Query Interval in the Encapsualator specifies the time interval of data to input into the Encapsulator from the data source.