HPIUM Basic Understanding- Session 3

- Presenter: Ujjwal Barman
- Date: June,2009
- Duration: 1 day

Agenda

- IUM Introduction
- IUM Basics
- IUM Architecture
- IUM Components
- IUM Error Handling Overview

HP Internet Usage Manager (IUM)

HP Open View Internet Usage Manager (IUM) is a flexible, scalable, open standards–based product for deploying Convergent mediation for both legacy and IMS (IP Multimedia Subsystem).

Deployed in wireline, wireless and broadband networks to support all voice and data services for prepaid, post-paid and real-time online charging.

IUM Advantages

- Scalability – carrier grade supporting multiple networks
- Reliability – collectors can run autonomously. IUM can also be deployed on a HA
- Extensibility – IUM can easily extend using SDK/PDK
- Manageability – supports a distributed collector architecture
- Security – provides an optional mechanism to authenticate, authorize and audit administration and configuration activities by users
- Flexibility – provides readymade general-purpose configurable collectors to support the various needs of billing, marketing, and operations management systems.
- Open Standards – IUM is 100% Java and user CORBA as the transport layer.

Convergent Mediation

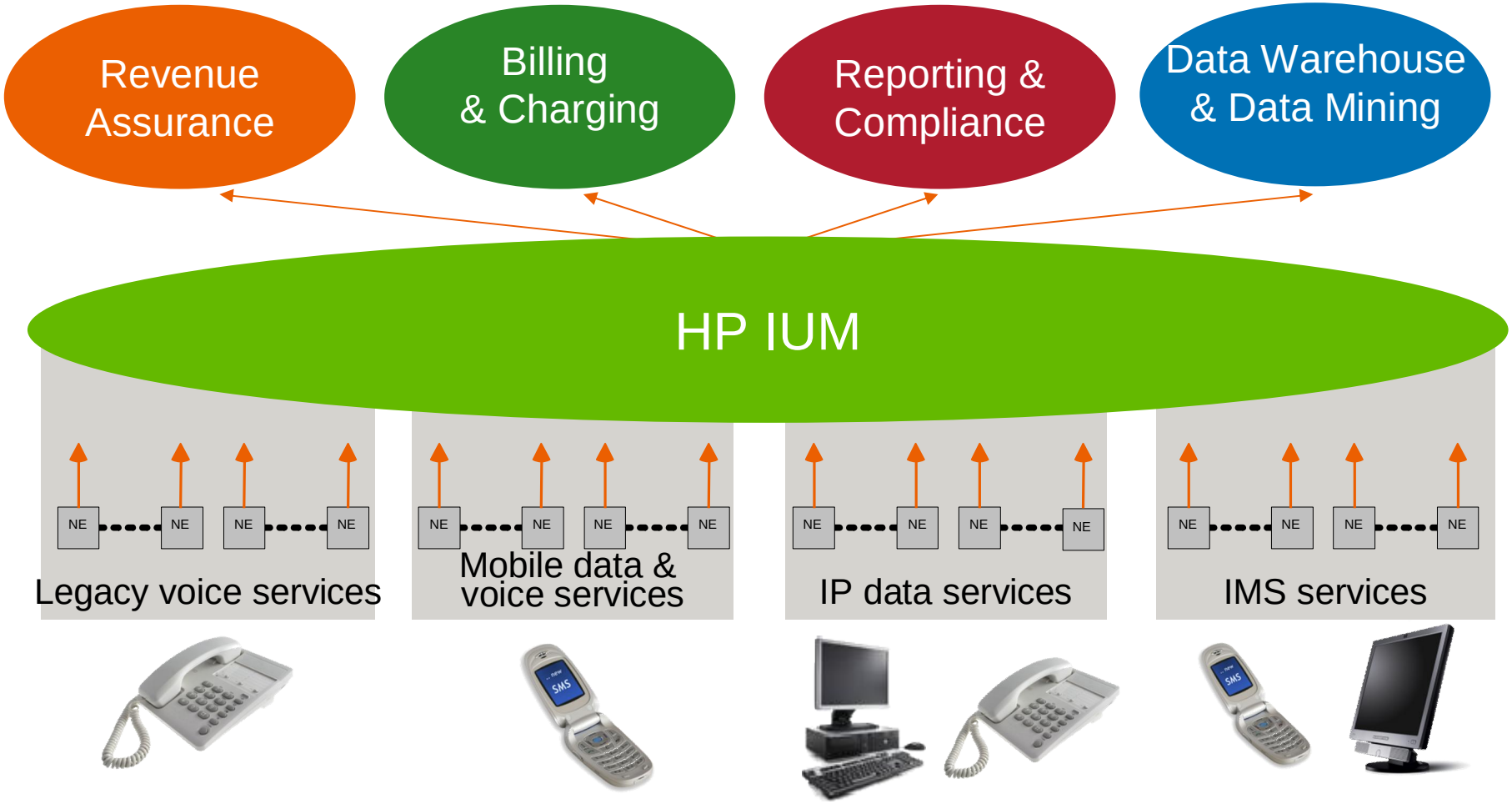
A process by which network usage data is collected, validated, aggregated, corrected and converted into billable business statistics.

HP IUM software supports this definition by enabling value-added billing, capacity management and subscriber analytics to develop strategic and profitable marketing programs

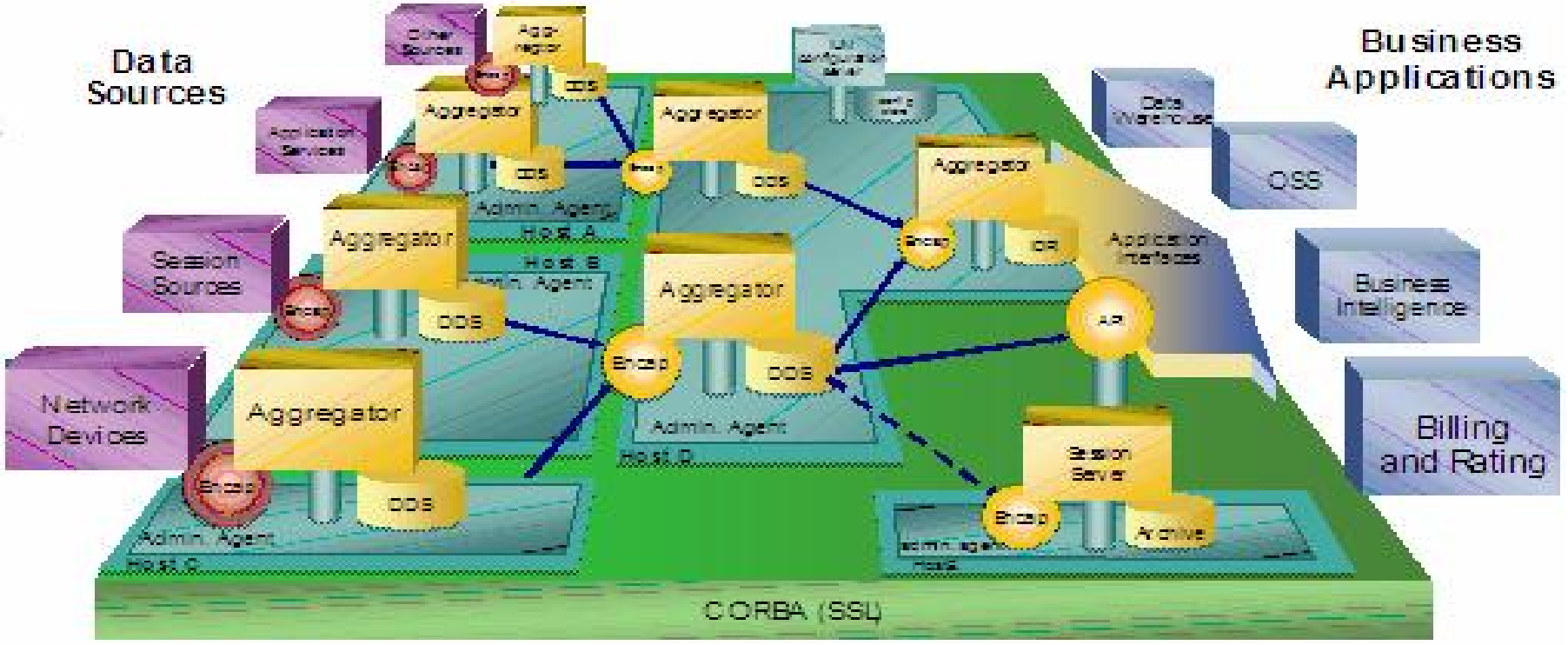
Example IUM use cases

Network/service type	Mediation use case
voice and broadband services (PSTN, ATM, Cable, DSL)	usage-based billing, interconnect billing
internet data center, web hosting, application hosting	bandwidth sales, capacity planning, quality of service
2.5G & 3G mobile data (GPRS, CDMA, UMTS, I-mode)	content billing, customer analysis, prepaid services
internet telephony, VOIP, VPN, multimedia	premium services, usage-based billing
circuit-switched voice	convergent billing mediation
storage/ server resource usage	cost allocation, capacity planning

One Platform for Convergent Mediation



HP IUM Architecture



IUM and CORBA

- CORBA = Common Object Request Broker Architecture; an industry
standard to allow distributed applications.

- This allows IUM to have collectors distributed across multiple machines, but managed as a single **deployment**.

- IUM uses an OpenSource CORBA product called JacORB 1.4.1.

IUM Data Collection

Transfer Mechanisms

- local file
- FTP
- HTTP
- FTAM
- SNMP
- GTP' (3GPP)
- JDBC
- RADIUS
- UDP (eg.netflow)

Input Formats

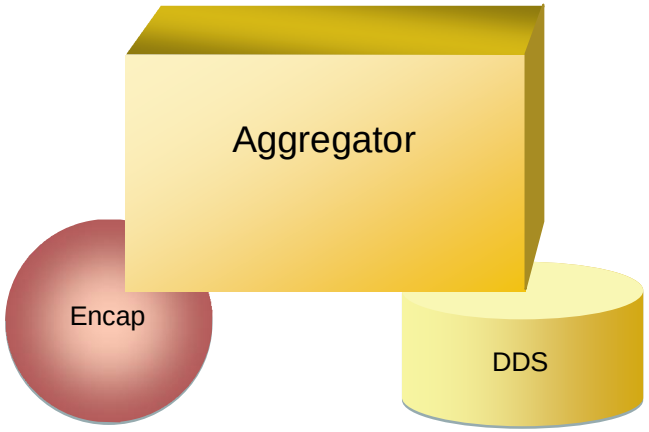
- ASCII
- ASCII delimited
- ASN.1/TLV
- TAP3 & RAP
- XML
- IPDR
- XDR
- SNMP v2 & v3, MIB-II, RMON2
- Cisco RTT MIB
- GPRS CDR
- CDR
- RADIUS
- Netflow, sFlow
- P-Cube RDR
- web, ftp, dhcp, proxy, radius logs
- vendor proprietary formats
- binary formats

Data Source Types

- AAA(DIAMETER/RADIUS)
- access servers
- databases (JDBC)
- DHCP
- DNS
- DSLAM
- firewall
- FTP Server
- GGSN
- LDAP
- mail
- MGC
- Probes
- proxy server
- router
- session

IUM Components

- **Encapsulator**
- **Aggregator**
- **Datastore**



IUM Data Processing,Output & Data Delivery

Aggregator

- aggregate
- validate
- filter
- correlate
- split
- add
- merge
- calculate
- perform lookups
- monitor thresholds
- generate alarms
- run external scripts

Delivery Mechanisms

- Delivery Agent
- FTP
- JDBC
- Local file
- SNMP trap
- HTTP

Output Formats

- relational database
- ASN.1
- binary files
- fixed or delimited ASCII
- HTML
- XML
- IPDR NDM-U 3.0
- AMA BAF
- EMI
- JDBC compliant external database table
- Proprietary formats

IUM Collector

- The primary component of IUM is the “Collector”
- Is a Java process that captures, normalizes, aggregates and correlates usage events as per rules defined
- IUM provides several pre-configured collectors with the product
- A collector can also be configured as a “Correlator” – to correlate data across multiple collectors to finally derive combined meaningful data
- Collectors can be grouped to form a “Collector Hierarchy”
- The hierarchy captures and processes multiple streams of input records generated by several data sources
- The collector hierarchy aggregates and processes the captured data and delivers consolidated records to downstream

What types of network data can be collected ?

IUM can consume virtually any network data:

- Level 2/3 network devices - Routers, switches, gateways ...
- Session sources - Network authentication/session services
- IP services - Email, web hosts, VoIP, VPN, ...
- Other services - SNMP devices, custom sources, ...

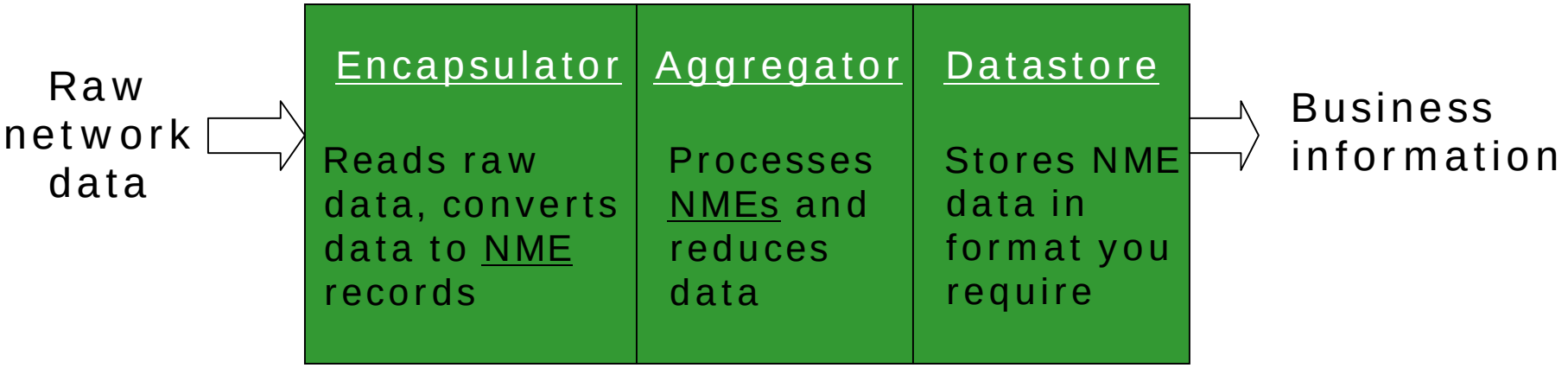
What information can be generated ?

IUM can generate data for virtually any application:

- Billing Systems - Based on usage of resources
- Fraud Management and Revenue Assurance – for fraud detection/correction and identifying potential revenue leakage
- Strategic Marketing - For competitive product positioning
- Capacity Planning - Anticipating subscriber behavior
- Data Warehouse and Data Mining - For in-depth business intelligence

Collector Components

A collector has three Components:



NME = Normalized Metered Event

What is an NME ?

NME stands for “Normalized Metered Event”.

NMEs are internal records used only by IUM. They provide a common data structure.

NMEs are created in the Encapsulator, passed to the Aggregator and stored in the Datastore. They may also be passed onto other collectors.

NMEs can be used to represent any kind of data:

- Session Events (session start/ intermediates /stops)
- Usage Events (duration, bytes of traffic, services accessed)
- Resource Events (disk space used)

Example Usage based NME

SrcIP	DstIP	NextHop	Numbytes	Start Time	End Time	SrcPort	DstPort

What is NME Schema?

- NMEs are composed of fields or Attributes that correspond with the various fields of some 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 LIST of all available NME types.
- The Encapsulator and Parser populate the NME with data that corresponds to the fields (Attributes) defined.
- You will need to add to the NME Schema to use new Attributes

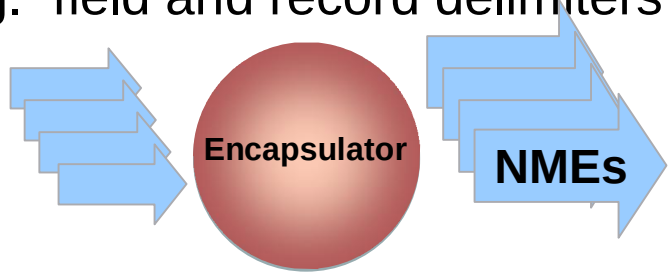
NME Schema

```
[/NMESchema/demo/BSCS]

Attributes=BSCS_MappingReferenceString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_StreamDescriptionString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_ServedPartyMsisdnString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_ServedPartyNumberingPlanString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_ServedPartyTypeOfNumberString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_ServedPartyImsiString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_ServedPartyImsiNumberingPlanString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_NetworkElementIdString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_NetworkElementNumberingPlanString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_OtherPartyNumberString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_OtherPartyNumberingPlanString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_OtherPartyTypeOfNumberString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_DialledDigitsString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_DialledDigitsNumberingPlanString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_DialledDigitsTypeOfNumberString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_RoutingNumberString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_RoutingNumberNumberingPlanString,com.hp.siu.utils.StringAttribute
Attributes=BSCS_RoutingNumberTypeOfNumberString,com.hp.siu.utils.StringAttribute
```

What is the Encapsulator ?

- The Encapsulator:
  - Reads data from a network data source or another collector
  - Converts input data into an internal record format called an NME
  - Determines when the data (NMEs) is stored in the datastore in a time-based model. Also determines when other collectors are queried.
- You may need to configure a parser depending on the input structure of the data, e.g. field and record delimiters



Reads data & converts to NMEs.
Uses a Parser for reading files.

Encapsulator

```
2417
2418
2419  [/templates/custom/msc_leaf/Encapsulator]
2420
2421  ClassName=RecordEncapsulator
2422  CleanupOnParseError=true
2423  CleanupOnRecordError=true
2424  CollectorLeafId=<Leaf collector identifier>
2425  Description=Processes event records from a file-based or record-based data source
2426  EndOnParseError=false
2427  EndOnRecordError=false
2428  PostParsedRules=010_Initialize
2429  PurgeOnParseError=true
2430  PurgeOnRecordError=true
2431
2432
2433  [/templates/custom/msc_leaf/Encapsulator/010_Initialize]
2434
2435  AdornOps=set MSC_ErrorCodeString 0
2436  AdornOps=set MSC_RawRecordString ""
2437  ClassName=AdornmentRule
2438  Description=Initialize values
2439
```

What is the Parser ?

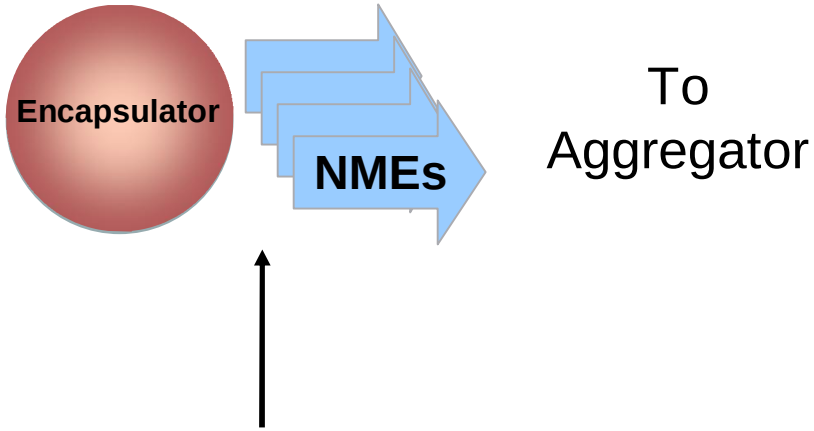
- The Parser is part of the Encapsulator. It translates raw incoming data records into fields that can be placed in an NME. For example, Motorola switch, Radius logs.
- A Parser is typically required whenever the data source is not another collector – when the data source is a network device or file.
- A Parser can also contain the following:
  - A Pre-processor - typically used to filter data during parsing.
  - A Post-processor - operates on data after it is parsed. For example, NMEAttributeValidationPostProcessor.
- Parselets - used to operate on data within fields. For Example, Dx200BCDLongParselet

Parser

```
2446
2447  [/templates/custom/msc_leaf/Encapsulator/Parser]
2448
2449  ClassName=TypedMuxParser
2450  Description=Delegates parsing operation to one of the sub parsers based on record type.
2451  Description=Input record type is treated as Hex Values and IgnoreTypes/MapParserType as corresponding decimal
      values
2452  IgnoreTypes=0-0
2453  IgnoreTypes=6-7
2454  IgnoreTypes=10-16
2455  IgnoreTypes=19-22
2456  IgnoreTypes=24-35
2457  IgnoreTypes=38-40
2458  IgnoreTypes=42-99
2459  MapParserType=MOC,1
2460  MapParserType=MTC,2
2461  MapParserType=FORW,3
2462  MapParserType=ROAM,4
2463  MapParserType=SUPS,5
2464  MapParserType=SMMO,8
2465  MapParserType=SMMT,9
2466  MapParserType=POC,17
2467  MapParserType=PTC,18
2468  MapParserType=UCA,23
2469  MapParserType=COC,36
2470  MapParserType=CTC,37
2471  MapParserType=USSD,41
2472
2473  [/templates/custom/msc_leaf/Encapsulator/Parser/COC]
2474  ClassName=OffsetParser
2475  Description=Parses COC record
2476  Attributes=MSC_RecordLengthHexString,0-1
2477  Parselets=MSC_RecordLengthHexString,Dx200StringParselet
```

What is a Flush Policy?

- The Flush Policy is part of the Encapsulator.
- It is a Time Interval which controls when NMEs are moved from the Aggregator to the Datastore.



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

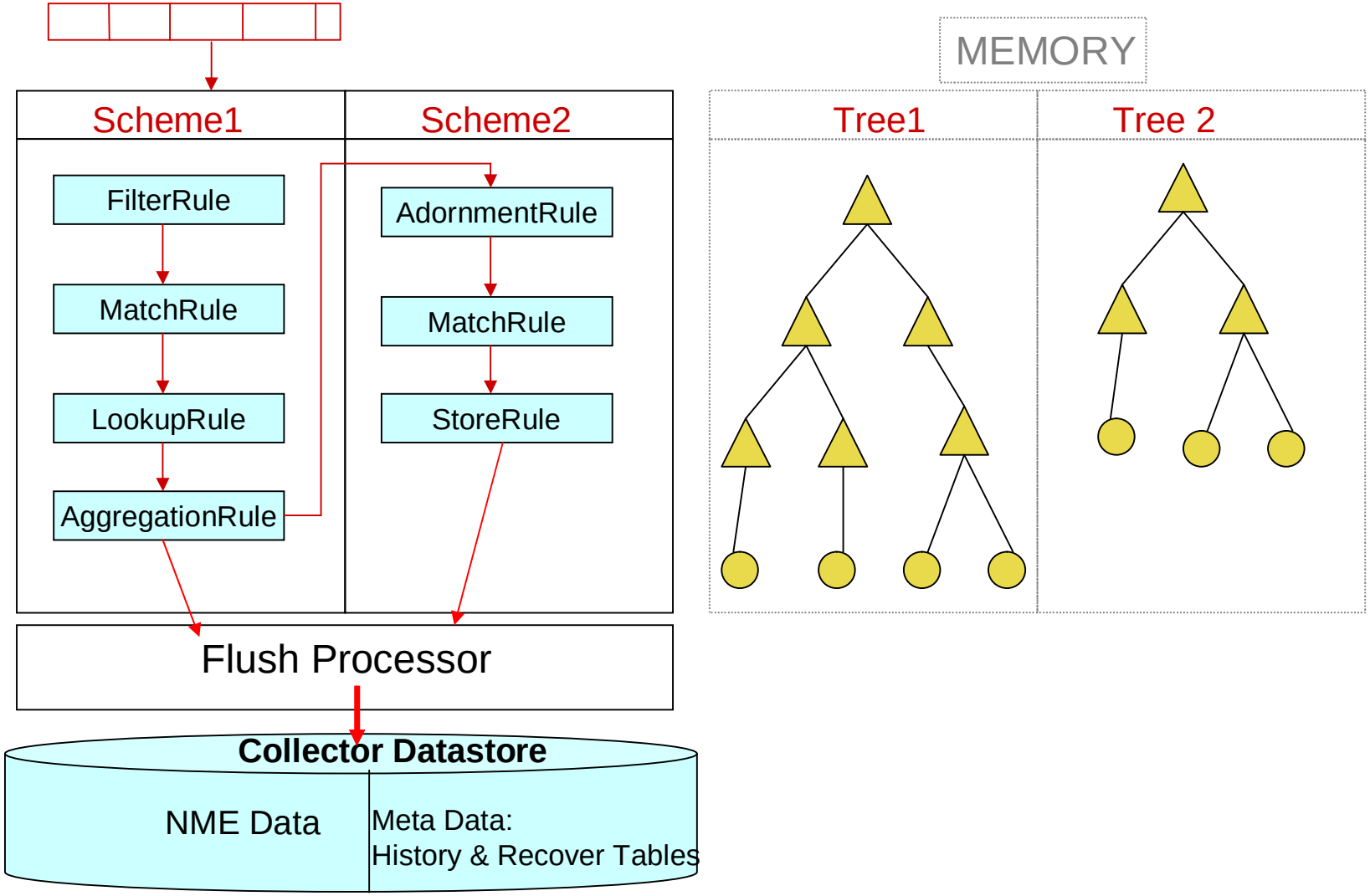
Flush Policy:

```
2439
2440
2441  [/templates/custom/msc_leaf/Encapsulator/FlushPolicy]
2442
2443  ClassName=TimeFlushPolicy
2444  ClockInterval=1m
2445
2446
```

What is the Aggregator ?

- The Aggregator processes NMEs using user-defined rules to filter, match, consolidate data.
- The Aggregator processes rules in memory by building a Tree containing NMEs.
- Rules are used to construct Rule Schemes. The rules control how a rule Tree is constructed (nodes and branches) and how NMEs are manipulated and stored as they pass through the Tree.
- The leaf nodes of the Tree are typically aggregated NMEs ready to be flushed to the datastore

Aggregator Architecture

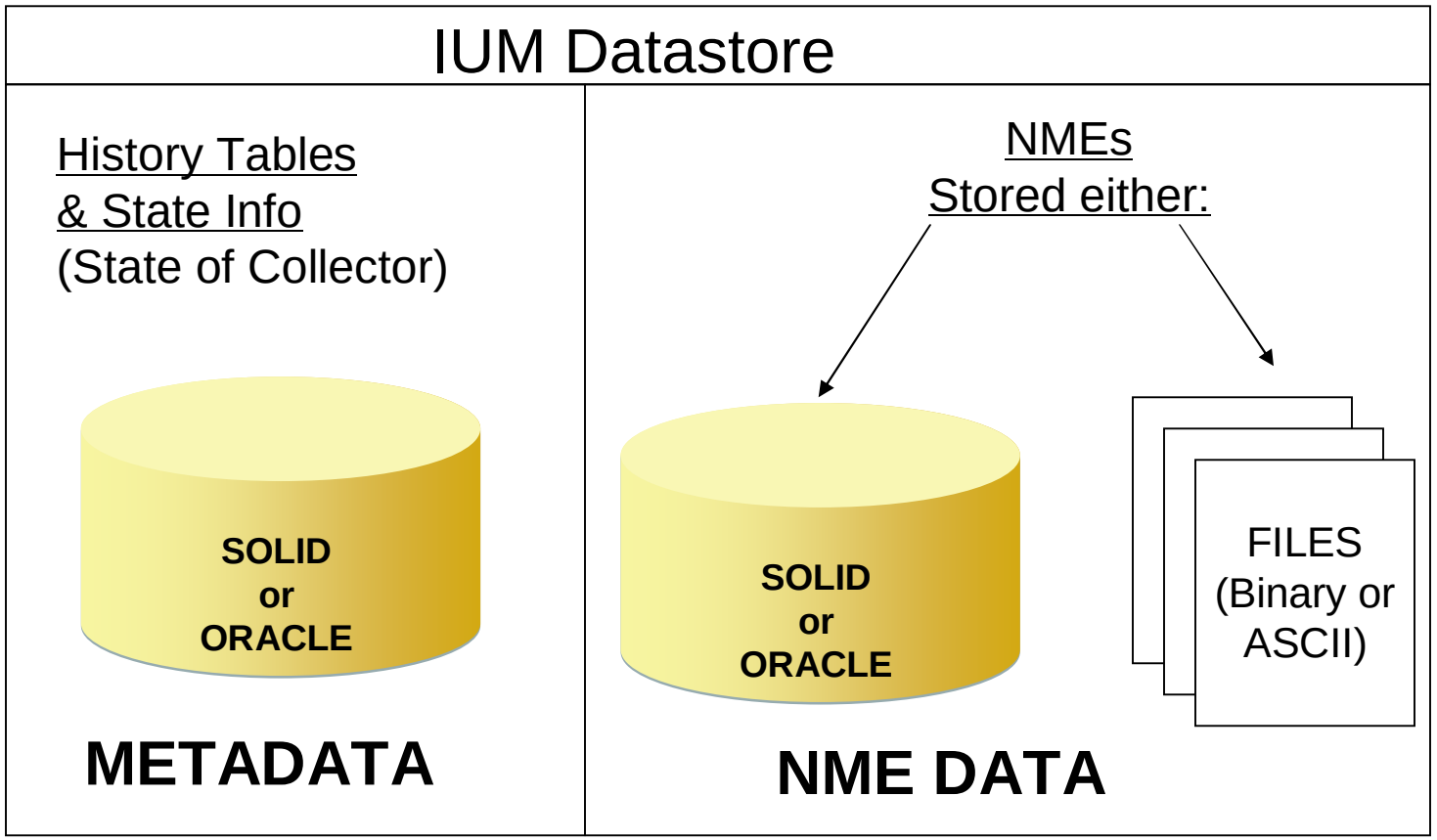


What is the Datastore ?

Every collector has its own datastore. It manages data storage.

How data is physically stored can be in many different ways.

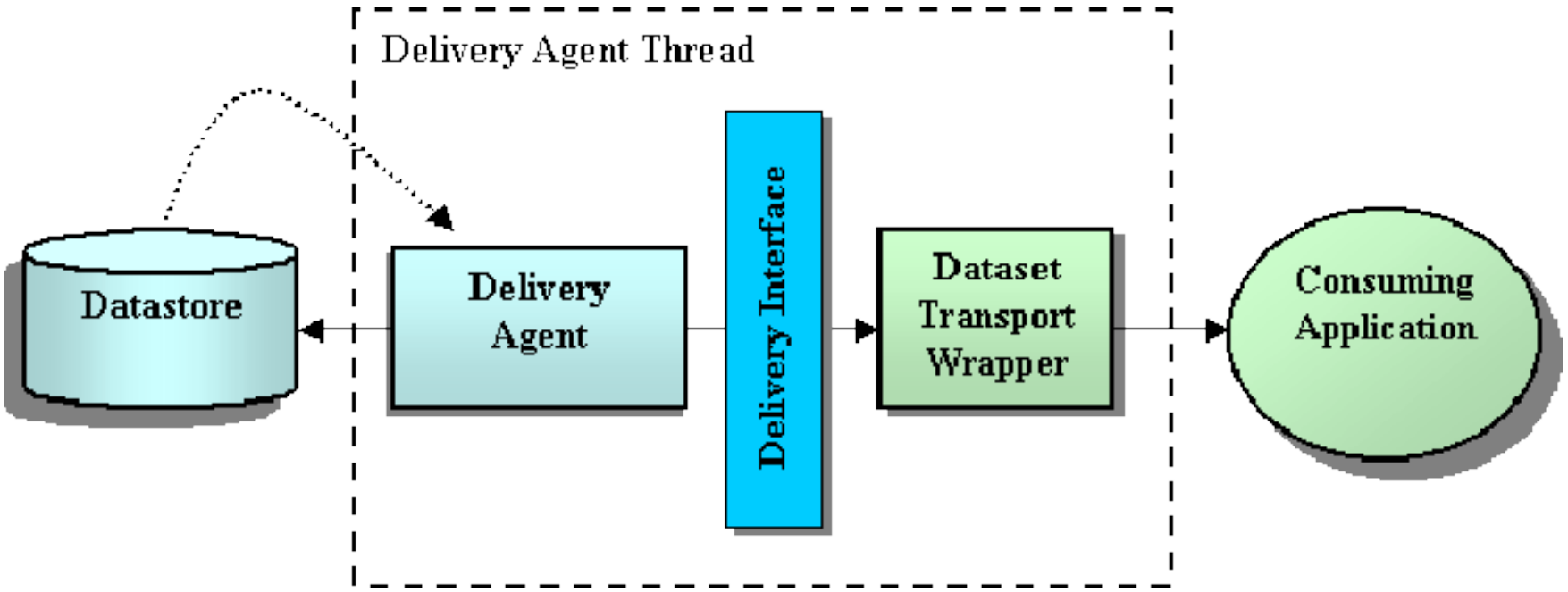
The datastore manages 2 types of data:



Data Delivery Agent

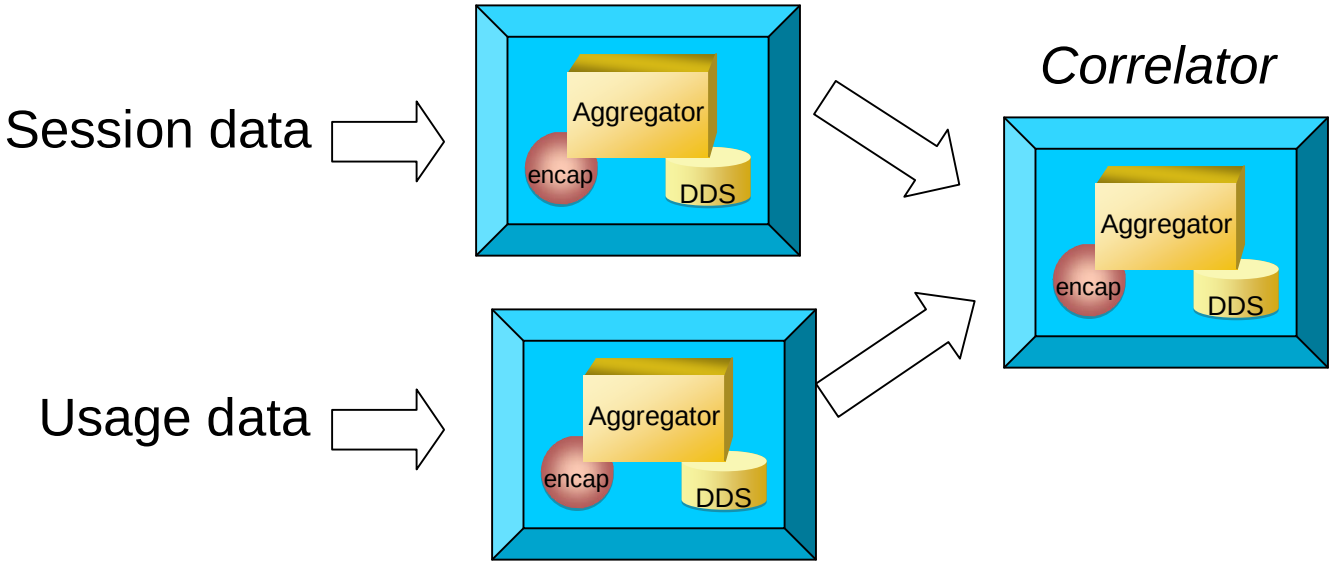
- The Delivery Agent is a sub-component of the Datastore that is responsible for delivery of data to a consuming application.
- A Delivery Agent runs as a separate thread off of the collector process.
- The Delivery Agent will track the flushes that it has successfully processed and pass them onto a consuming application.
- The Delivery Agent doesn’t know (or care) what the consuming application is.

Data Delivery Agent …Continued



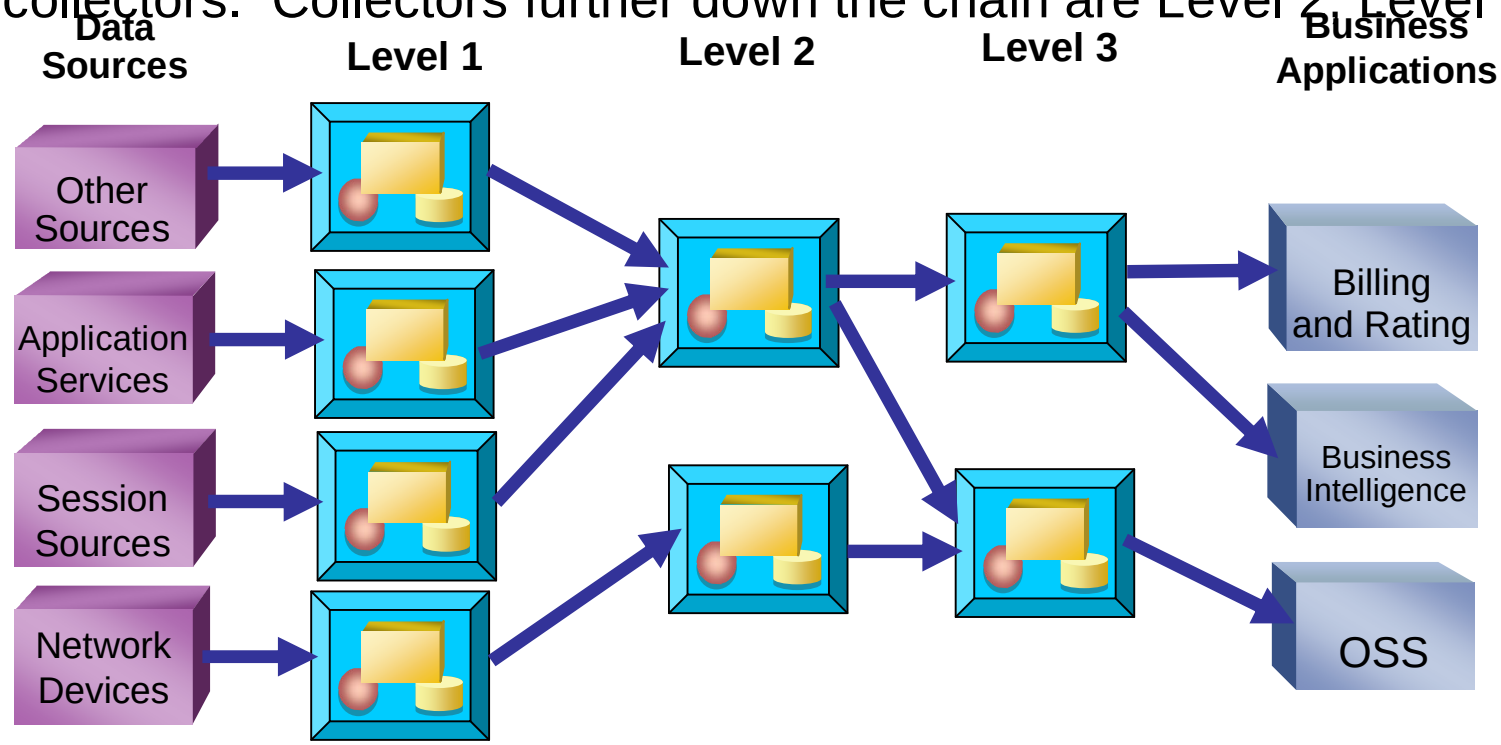
What is Correlation?

- A Correlation Collector is a specific type of Collector that associates usage and session data from other Collectors.
- Correlation involves a specific rule - CorrelatorMatchRule.
- The CorrelatorMatchRule first sorts session NMEs and then locates the appropriate session in the Tree for the inbound usage NMEs.
- The Correlator uses a polling mux encapsulator.



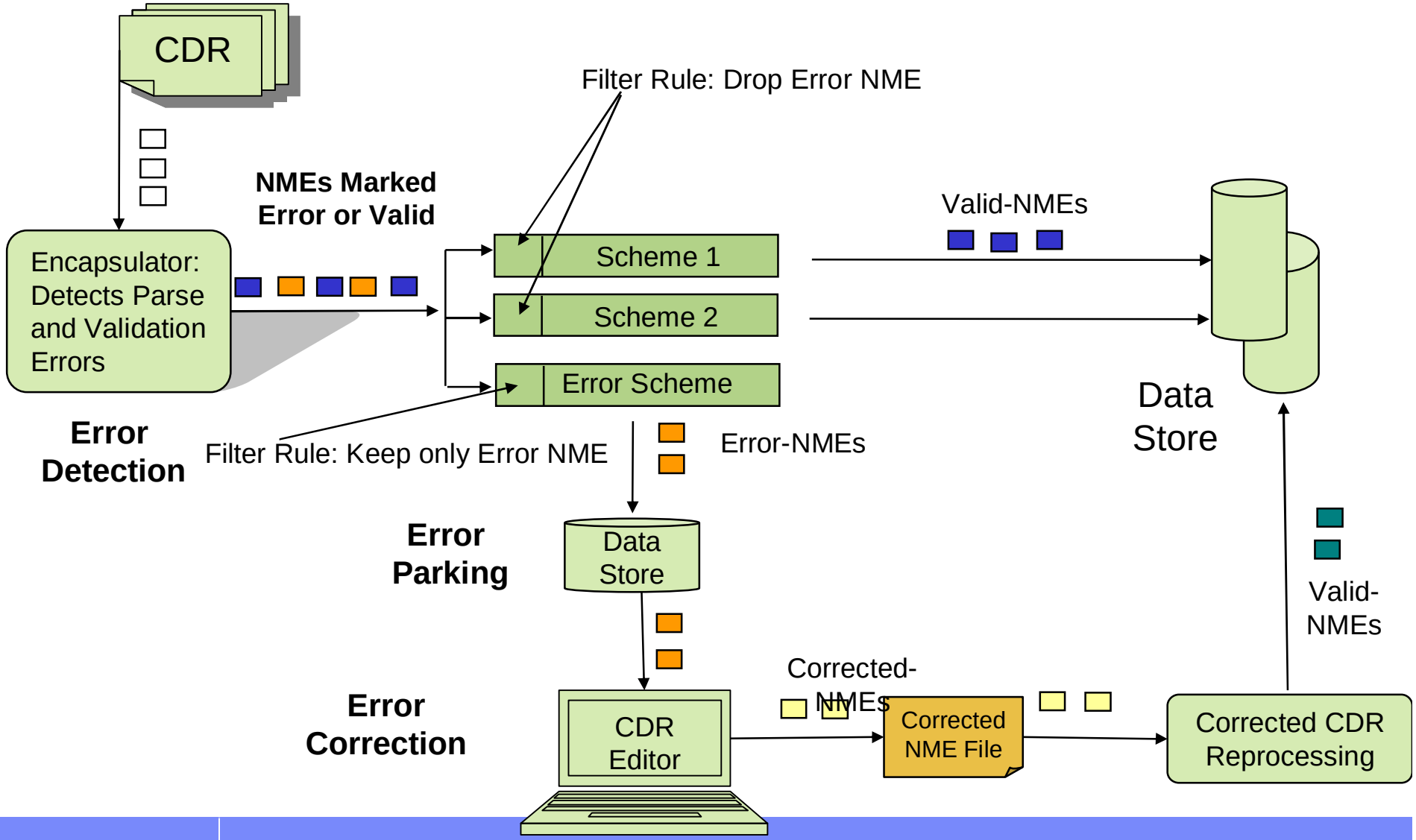
Collector Processing Chains

- Collectors are chained together to perform multiple levels of data processing.
- Collectors next to network data sources are known as Level 1 collectors or Leaf collectors. Collectors further down the chain are Level 2, Level 3, etc.



Examples of Level 2 and Level 3 collectors – correlators, reporting collectors, error handling collectors, etc.

CDR Error Handling Overview



Corrected CDR Reprocessing

- Solution Overview:
  - Re-processing collectors are configured to automatically collect the files that have been corrected by the CDR Editor.
  - Once the re-processing collector has re-processed the correct NMEs they can be sent to another collector which can merge the correct NMEs with the valid NMEs from the first collector.

IUM Sample References

- Amena, Spain
- Blixer, Italy
- Central Telegraph, Russia
- Colombia Movil
- Cox Communications
- Fastweb, Italy
- Italtel
- Korea Telecom
- KT Freetel, Korea
- LGT Korea
- MTS Russia
- NEC
- Retevision, Spain
- Rogers Cable
- Jiangsu Mobile, China
- Lumena, Spain
- Radiomovil
- Retevision Movil
- SBC
- Telecom Italia
- Telstra, Australia
- Telefonica, Spain
- TeleGreenland
- Telstra, Australia
- Telecom New Zealand
- Telus, Canada
- Vodafone D2 Germany
- Vodafone Greece
- Vodafone Spain
- Vodafone U.K.
- Du,Dubai

References

- **IUM user guide**
- **IUM Command Reference.**
- **IUM IUM Component Reference**
- **IUM Self learning kit**
- **IUM troubleshooting guide.**