MDM

* **3 implementation styles of mdm-**

1. **Registry**- Collect data from different sources to create an index.  
   Records are linked into entities, one index may point to multiple records across systems.
2. **Transactional**- Depending upon transaction, match and merge or detect duplicates.
3. **Hybrid**

* **PME- Match engine working on probability**
* **Integrating mdm with other tools, 2 modes are available-**
  + **Real-time mode-** Any change in MDM is reflected in the integrated tool immediately. Come in many forms(RMI, JMS, Webservices, JMX etc)
  + **Batch mode- Scheduler** is used to reflect the MDM transactions after a particular period of time.
* **Attribute- Info attached to a business object.**
* **Component Methods-**
  + **CRUD- Create, update, read, delete**
* **MDM Services-**
  + **Fine grained- One** Transaction on One object only or two objects having a has-a relationship. The services contain CRUD services(add, update and get).
  + **Course Grained-** More than one objects affected or more than one transactions take place on a single object,
* **TCRM- Tag denoting transactions**
* **Business vs admin services-**
  + **BS is transaction reference guide and manages master data, services performed on business objects. TAG- TCRM**
  + **AS is common transaction reference guide and has mdm server information, services on admin business objects.TAG- DWLV**
* **WHAT TO KNOW ABOUT SERVICES-**
  + Parent Object( or parameters for inquiry services)- Parents in terms of hierarchy
  + Service Extensions(Transaction)
  + Service Extensions(Action )
  + External Rules
* **Transaction vs Action**
  + **Transaction** is a user-defined part of script. You can name any number of your script as one transaction. This is a kind of logical division of your script.   
    For example, you can have the following transactions:   
       - Login   
       - Checking if new messages are avalable   
       - Getting new messages   
       - Logout
  + **Action** is a physical part of script - this is a separated file, located on file system.   
    Default script contains of three parts - vuser\_init, Action, and vuser\_end - an initializing, a test itself, a finalizing correspondingly. Each one of these parts is an action.   
    An action can contain zero or more transactions.
* **Access Control Group-** 
  + **Authentication**
  + **Authorization**
  + **Encryption**
  + **Auditing-** Tracking user activity
* **Defining Groups through LDAPs**-
  + **LDAP** (Lightweight Directory Access Protocol) is a software protocol for enabling anyone to locate organizations, individuals, and other resources such as files and devices in a network, whether on the public Internet or on a corporate intranet.
  + Groups are created and added in WorkBench which defines the “Authorization” for the groups. Permissions are then published to the Master Data Engine.
  + Workbench- Platform to extend MDM. For eg, to write more validation rules, match-merge rules.
* **Group tabs allow yout to specify following permissions-**
  + Set access to specific sources or attributes.
  + Allow users to perform specific API functions
  + Allow access to web reports and the Inspector Data Manager function
  + Allow access to composite views(Providing a complete picture of a particular member).
  + Allow access to view relationship attributes.
* **Secure Socket Layer(SSL) Encryption-**
  + Provides encrypted form of data transmission .
  + Hanshake method used.
* **ROV(Rules Of Visibility) and Data Entitlements-**
  + Visibility rules- which elements or instances of elements are visible to which users.
  + Data Entitlements- Which elements or instances of elements can be added or updated
  + Access Tokens- Row level security handling who can see and modify certain party or account information.
* MDM stores operational, standardized data which is generally not used for business intelligence. However DW contains large amount of historical data with no particular standard and is used for trend analysis.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# DAY 2

* **Service Oriented Architecture-**
  + **CONSUMERS:** User interface to communicate the demand, available in the form of different forms of channels, B2B and portals.
  + **BUSINESS PROCESS:** The flow of request and the entore business transaction.
  + **SERVICES:** Services the consumer wants to avail. Can be atomic(single service) or Composite(Many services at once). This is where the consumers invokes the service.
  + **SERVICE COMPONENTS:** Details of the Services available for the user to be used.
  + **APPLICATION SERVICES:** Can be Packaged, Custom or OO(out of box) application. This is the platform where the services are executed. In MDM, custom applications are used generally. CRM or ERP come under this layer.
  + **DATA REPOSITORY**
  + **INTEGRATION:** Information flows from MDM to another tools through ESB(Enterprise Service Bus).
  + **QUALITY:** Security, Management, Monitoring
  + **GOVERNANCE:** Monitoring of People, processes and data.
* **Information as a service provides:**
  + Access to master data through a large set of well defined SOA services.
  + Consistency in the view and packaging of the data.
  + Consistency in the rules applied to the data.
  + A single point of maintenance.
* **MDM Server manages the following types of master data and the relationships aming them:**
  + **P**arty
    - Agent
    - Customer
    - Supplier
    - Employee
    - Dealer
  + Product
    - Product
    - Product Bundle
    - Service
  + Account
    - Contract
    - Agreement
    - Reward Program
    - Financial Account
    - Transaction
  + Location
    - Address
* Catalogue-> Hierarchy-> Categories-> Product-> Attributes
* **MDM Data Model**
  + MDM DocStore- Where data goes first before going througfh further stages.
* Qualities OF Master data-
  + Domain Centric
  + Application Agnostic
  + Process Neutral
  + Consistent in vocabulary and structures
  + Enforced Quality- Proactively
  + Industry agnostic
  + Technology agnostic
  + Configurable and Extendable
  + Alignment with Industry Models And Assests
* **Approxiamtely 900+ services**
  + **Inquiry**
    - **Get.xx**
  + **Search**
    - **Search.xx**
  + **Persistent**
    - **addPerson**
* **Granularity:**
  + **Course-grained:**
    - **addPerson=addPersonName+ … + addPartyAddress + … + addPartyIdentification…+…**
  + **Fine-grained**
    - **addPartyIdentification**
* **3 Modes of Operation:**
  + Real Time
  + Batch Mode
  + Scheduled
* **Styles of Operation:**

**(Analytical Focus)**

* + **Consolidation Style**
    - Matches and physically stores a consolidated view of master data.
    - Consolidate master data to share with a target system or as a preparatiom for transaction hub deployment.
    - Updated after the event and not guaranteed up to date
    - Not used for transactions but could be used for reference.
    - For reporting, analysis and central reference **(Operational Focus)**
  + **Registry style**
    - Matches and links to create a skeleton system of record
    - Virtual, consolidated view, assembled dynamically and often read-only
    - Mainly for real time central reference
    - Physically stores global id and
  + **Coexistence style**
    - Similar to consolidation, has index and gives a single view except that no external repository is accessed.
    - Updated after some event and not guaranteed up to date.
    - Can be a logical starting point for many projects
    - Publishes the consolidated view.
    - Not for transaction, but used for reference.
  + **Transaction style**
    - Requires up front strategic planning
    - No index and no repository
    - Usually chosen because a long-term enterprise-wide MDM strategy is in place.
    - Faster and apt for achieving short-term goals
* **MDM ARCHITECTURE**:
  + **MDM Consumers**
  + **Request Framework**
  + **External Components**
  + **Extension Framework**- Mechanism For extending Service behavior and data model using Workbench
  + **MDM Core**
    - Controller Components
    - Business Logic components
    - Utility components
    - Suspect processing components
    - Run query
    - Code tables- Rule data
    - Operatinal structure: data
    - History structure: Historical data
  + **IM/AIM integration**
    - DataStorage
    - Quality stage
    - FileNet
    - iLog jRule- Create your rules here
  + **Common Components**
    - Externak Rules
    - External Validation
    - Event Manager
    - Notification
    - ROV
    - Metadata
    - Matching
    - Caching
    - Service Activity Monitoring Facility
    - Performance tracker
    - Error messaging
    - Logging
    - Configuration Manager
    - Search
    - Standardization
    - Task Management
    - Specification
* **REQUEST RESPONSE FRAMEWORK**
  + **DWL Service Controller**Transaction controller**:** services that can be rolled back, I,e, persistent services

Finder controller: Services that can’t be rolled back.

* + **Request Handler**
  + Parser: Responsible for parsing the incoming request and converting it into a format understood by the target enterprise application
  + Constructor: Performs the opposite function of parser, and converts the data format returned from the target application into a format to be returned to the client.
  + EntityObject: A subpart of Business Object and contains information regarding the object but is not the real object.
  + **Business Proxy**
  + Persistent Transaction- if it can be rolled back
  + Non- persistent transaction- it can’t be rolled back
  + Service controller-> Request Handler\_> Business Proxy(DWLTxnBP) | --------> Core MDM server or
* **PARTY:**
  + Logical Primary key- PartyId
  + Physical Primary Key- cont\_id
  + Logical Entity: TCRMPartyBObj
  + Database table –CONTACT
  + Standardized fields start with s: s\_given\_name\_one
  + Phonetic fields start with p: p\_given\_name\_one
* **SOAP**- Simple object access protocol:
* **Service Invocation Protocols:**
  + RMI-IIOP
  + JMS
  + WebServices
  + JMX
  + Default readers/writers use files ( For invoking services in BATCH mode)
* **MDM services are transactional:**
  + **Invoking any Infosphere service is a transactional operation.**
  + **Services over RMI-IIOP and JMS protocol are stateless services**
* **Receiving the request: DWL ServiceController**
  + RMI invoked using processRequest(Hashmap context, Serializable processRequest
* **3** **Modes Of Invocation Of Services:**
  + Web Services mode:
    - Supported standards:
  + Messaging Adapter:
    - Uses a message-driven bean
    - If you use webservices, no response is received, only the output.
    - But in the case of message driven bean like MQ- portal or JMS, 3 types of responses can be obtained: SUCCESS, FAILED or BACKOUT.
  + Batch Requests:
    - Batch Processor:
      * Handles each record in its unit of work.
      * Multi-threaded execution
      * Can dynamically adjust threads.
    - Pluggable readers/writers:
      * Reader responsible for reading the batch input and returning one record at a time.
      * Writer is responsible for writing the batch.
      * Writers can be chained, Like Chained FileWriter, SuccessWriter.
    - WAS XD Batch Processor:
      * Driven by an XJCL batch job
    - Common usage includes **initial** **loads**, **delta** **loads**, extracts.
      * INITIAL LOAD- initial loading of data
      * DELTA LOAD- **In case of delta, first load should be a init, which tells the source system to keep the track of changes/new records in order to send only these changed/ new records into BW which flows as delta.**
* DWL RequestHandler:
  + Invoked by DWL ServiceConmtroller and passed the Request
  + If ASI\_Request is defined:
    - DWLRequestHandler used ASI request adapter to parse the request
  + IF ASI\_Request is not defined
    - ParserFactory determines the parser from the DWLCommon.properties  
      Parser.<TargetApplication>.<Parser>
    - DWLRequestHandler invokes the parseRequest
* **PARSER**(WithoutASI):
  + **Constructs a transaction object**, based on the type of request:
    - A transaction object can be an **inquiryTransaction , searchTransaction** or **PersistentTransaction.**
    - Each request type defines which transaction object to create in the CDBusinessTxnTP table
  + For Search and Persistent, constructs the BusinessObject(Bobj).
  + Bobj constructs an Entity Object(Eobj).
* **ADAPTIVE SERVICES INTERFACE**:
  + Parsing and generation of response is done by ASI Adapter
* **CONSTRUCTOR**
  + **Constructs the response to be returned to the caller.**
  + Define in properties file in the format of:
    - Coomstructor.<TargetApplication>.<Constructor>
  + Defualt Constructor:
    - Infosphere MDM XML Constructor
    - Composite XML Constructor( used in composite request framework)
    - If ASI\_Response defines, ASI Adapter is used.
* **2 steps of security in MDM-**
  + Authentication: are you who you say you are?
  + Authorization: Are you allowed to do that?
* **MDM Security steps:**
  + **End user authentication**- Log in the pc and open an application
  + **MDM Server connection** – Establish a connection with the server
  + **System user authentication**- Log into that server which will determine the types of services you’re authorized to use
    - **System user handled by application server**
    - **2 security roles used by mdm server**
      * **ServiceConsumer**
        + All user/group authenticated by applicatrion server
        + Methods of MDM server EJBS grant access to ServiceConsumer
      * **ServiceProvider**
        + MDM Server(RunAs)
        + Bound to an username/password in user registtry
        + Username/password should be reserved for MDM server
  + **Request** **authorization-** Checking whether you’re authorized to make this request.
    - **Handled by Infosphere MDM(DWL RequestHandler)**
    - **Default Transaction authorization provider**
      * Infosphere MDM db stores info about security policy
      * A security manager administers the authorization data
      * The authorization data associates users and groups to the transactions that they’re authorized for.
    - **LDAP Transaction Authorization Provider**
      * USES JNDI to connect to the LDAP server
      * Uses LDAP search functionality to query the directory for a relaitonship between the transaction and the group or user
    - Plug in point
      * A custom transaction authorization provider
      * A custom user management provider
      * A custom authentication assertion parser to parse out user/group information
* Entity objects are detailed business objects.
* Partyname, partyaddress- are different business objects. But they do not have a unique identification number as they represent a single entity.
* XML contains details of transactions regarding following domains: party, product, location, etc.
* Adding a new party:
  + No metadata required
* Adding a product
  + Metadata required
* **Application Framework:  
  3 parts of a.f.**
  + **Vocabulary of a service**:
    - Information received in the form of request from the service and returned in the form of response.
    - Consists of **control structure** known as DWLControl(which controller to be used) and **DataPayload** represented as set of parameters for most inquiry services and business object graph for persistent and search services.
    - By convention, xxxBObj
    - Usually an abstraction of the corresponding database entity
    - Maps directly to XML schemas for RMI and WebServices interfaces.
  + **Behavior of a service**:
    - What time of a query it is; what type of
    - Services are implemented using a component based approach.
  + Pre/Post Processing of a service: The type of request obtained( type of object, transaction, etc) and the response generated ( in the form of xml)
* RR Framework -> Txn Controller/Finder Controller-> Business Component –(Bobj)-> BobjQuery-(Eobj)--> PureQuery-> ODS
* Business Component: Depending on the type of controller, from which the bobj is obtained, services are performed.
* **Role of DWLControl Object:**
  + Provide context and processing info abt the transcation.
  + DWLCOntrol is shared by business object and any of the child objects under it.
  + A subset of elemsents from DWLControl stroed in transaction audit log and can drive the conditions of when to execute behavior extensions
  + Name/value pairs
* Entity Object:
  + By convention, Eobjxxx
  + Strongly typed attributes
  + Maps to the physical data model
  + Most entity classes are annotated to map to the physical data model
* **BUSINESS LOGIC COMPONENT:**
  + Heart of Infosphere MDM
* **Bobj query framework**
  + Encapsulates core database access functionality
* **PureQuery**
  + Heart of persistence framework
  + Used for all its database access
  + MDM Workbench generates all the persistence code with the help of PureQuery
* **Operational Database**:
  + Contains operational data for master dtaa domains
  + Data retrieved and persisted using persistence framework
* **History Database:**
  + Replica of operational database’s schema with additional control fields for tracking history
  + Access for ‘Point in Time’ (PIT) or data-range inquiries
  + Populated using simple or compound data base triggers.
* **PureQuery Classes**

Eobj-> EobjxxxData-> EobjxxxDataImpl-> ODS

* + Eobj: Entity object contains database mapping and attributes
  + EobjxxxData: Interface class conatins SQL definitions; Has two parts: SQL and pattern
  + EobjxxxDataImpl: Implementation class generated by entity class
* **Pluggable persistent layer:**
  + Provides extra features like Pluggable id generators, smart inquiries, paginf and summary indicators(for eg, retrieve addresses for a given party only when we know that the party does have one or more than one addresses)
* **Pre-Post Transaction Framework**:
  + Part of application framework
  + Allows business transactions to inherit common behaviors.
* PreExecute() Transaction Logic:  
   **At the beginning of each transaction, the framework will”**
  + Populate Before Image for Update Controller level transactions
  + Handle standardization of objects
  + Handle multi time zone data attributes
  + Invoke internal and external validation
  + Invoke behavior extensions

**PreExecute() Business Component Logic:**  
At the Beginning of each businnes component, following will :

* + Invoke internal validation
* Chevk for redundant updates for Update transaction
* Invokde deleteRecord() on Business Objects

**PostExecute() Transaction logic**

**Invoke behavior extensions**

* **MDM SERVER EXTENSION FRAMEWORK:  
  2 main extension types:**
  + **Data**: Changes to the data model: Add more attributes the existing data
    - **Data Addition:** Adding new entities or subject area to the model. Also involves adding new supporting services to the product framework.
    - **Data Extension:** Adding new elements to the existing entity in the data model; Extension framework allows for these new elements to be persisted and retrieved by all existing services consuming the modified entity.
    - **Specs:** Define new attributes for Products, Parties, Contracts using **XSD**; Attribute values are stored in the database as XML.
    - **2 approaches to data extension persistency:**
      * Persist attributes to an existing table
      * Persist attributes to a new table
  + **Behavior**: Changes to service execution
    - **Service behavior:** Adding new behavior to existing services; Pre/Post framework
    - **Query behavior:**Change the query implementation for an xisted service
    - **Composite service behavior:** Pull existing services together to execute as one unit of work
      * **Customize business proxy**
      * **XML**
* **MDM WORKBENCH:**
  + Integrated set of Eclipse based tools
  + Code generation for:
    - Data extension
    - Behavior extension
    - Specs
    - Data additions
    - Service additions
    - Composite services
* **Generated Artifacts:**
  + Model driven
  + MDM Workbench generates artifacts from models
  + Supports incremental development
  + Preserves manual changes to code on regeneation

**Eg: EJBs, DDL database definitions, Web services definitions, RR XSDs**

* **Data objects:**
  + Entity: Entity addition
  + Entity Extension:Extending existing entities
  + Transient data object: Not persisted to DB, only used in request and response
  + Code table: Simply key/value lookup tables
* Entity Containment:
  + A containment is associated to reference
  + Extends the way the reference is processed
  + When an entity is queried, all the contained entities are also retrieved.
* Workbench transactions:
  + Persistence (Add, update)
  + Read –only(Serach, view)
* Query Extensions:
  + Permit customization of the SQL used to retrieve data
  + SQL exposed in the generated code and can be modified
* Data Standardization:
  + Java classes that implement the standardizer interface and thus standardize entities before they’re persisted in the database
* **Library objects**:
  + Predefined content that may be associated to an entity
* **Specs**:
  + A type of metadata used to define a data extension
  + Uses XSD to define structure
  + Does not change the database schema
  + Values of specs are stored as an XML Document
* Why specs?
  + Flexible: Create data extension without redeploying the server
  + Structured: Unlinke Miscellaneous values, Data attributes are well defined,
* When to use specs?
  + Short lifecycle: seasonal product
  + Domain: attributes apply only to specific entities
  + Time to market: Can be created faster than hard attributes created though extensions and additions to the data model.
* Where to add specs?
  + Product domain
  + Account domain
  + Party(Demographics)
* Behavior extension Java class:
  + Public class ExtensionBehavior extends ClientJavaExtensionSet{
    - Public void execute(ExtensionParameters param)
      * Behavior changes of we change the transaction type
* Famous Behavior extensions:
  + Rules of Visibility: Post Transactions
  + Data Entitlements: Pre Transaction
  + Smart Inquiries
  + EAS Feed
  + Party Compliance(Event Manager)
  + Terms and Conditions Evaluation(Event Manager)
  + High Value Customer(From the ZZ640 customer)
* XML Composite desired when:
  + No java skills or code is required
* Use Business Proxy when:
  + Performance is a factor
  + New MDM entity is added
* **Deterministic search**: Based on a unique parameter
* **Probabilistic search**: Match with every field and give a weight whose cumulative gives an output
* **EVENT MANAGER:**
  + A set of database tables, processes, and business rules within InfoSphere MDM that can track and react to system-defined or user-defined events.
  + Can be used to satisfy business requirements:
    - A customer cancels credit card.
    - A customer turning 65 years old.
    - A customer withdrawals funds from a tax-sheltered account.
  + Technical USE:
    - Suspect Duplicate Processing:
      * The application sends suspects to another engine for matching asynchronously after a transaction.
    - Evergreen Processing: When we try to load data in batch mode, i.e in bulk, suspect duplicate processing and standardization are done.
    - Acxiom: For standardization
    - Business data corruption
    - Compliance: If a user is complying to the assigned rules and whether trying to access he’s forbidden to do so.
  + **When are events detected?**
    - Transaction based
      * A customer cancels the credit card
        + At the postExecute() of an updateContract transaction
    - Time based
      * A customer turning 65 years old
        + Event manager invokes the scheduler to query thhe party to see if the party is 65 years old
    - Ad-hoc
      * A customer calls the CSR to request withdrawal
        + The CSR creates an explicit event.
  + Detected events are persisted and notification is sent out, if enabled.
  + **EM Architecture**:
    - **PROCESS CONTROLLER MODULE**
      * **Process Controller Internal:** Retrieves event categories and refers to event task
        + Invoked by a EventDetectionScheduleController to detect events from the ProcessAction table. In case of time based events, EventDetectionScheduleController makes a remote call to Process Controller Module.
        + Retrieves process actions that are passed Next Process Date
        + Creates Event Task for each Process Action
        + Place eventTask on queue.
      * **Process Controller**: Based on the event task, refers to DB.
        + Two ways to invoke:

Send Explicit Event Task

Send Business Object for event detection

* + - **Event Detection Module**
      * EventTask🡪 Entity Adapters
      * EventDetectorHelper
      * EventDetectorMDB-(Message driven bean)
      * EventDetector
      * NotificationSender🡪Pubsub
    - External Rules
  + **Types of Events:**
    - **Events informed about by clients( AD-HOC):**
      * Not processed by event manager
      * Stored in event table using addPartyEvent service
      * Event are used in proccessing other events
        + Eg- Event stores that customer is retired, therefore Party can withdraw from tax sheltered account
      * Other Services available:
        + getAllPartyOccuredEvents
        + getAllPArtyPotentialEvents
        + getPartyOccuredEvent
        + updateParyEvent
    - Scheduled by time or **TIME BASED** :
      * A scheduler is used to invoke the process Controller Internal.
      * Creates an EventTask for each ProcessAction that has passed its ‘next\_process\_dt’.
      * Retrieve the Business Object that will be processed from the Process COntrol
    - **EVENTS RESULTING FROM TRANSACTION**
      * Infosphere MDM invokes the processTask method on the Process Controller
        + Input incukdes the businessobject key (B partyId), Nusiness object and DWLControl
        + Optionally can provide ab explicit event detection to run using the eventTaskParameters
      * Event Manager Creates an EventTask for each Event defined for the BusinessObject
      * EventTask is Passed to the Event Detection for Processing
  + THESE RESULT IN:
    - Email generation
    - Database update
  + **Proces Controller:**
    - Retrieves event categories
    - Create event for each event category and place in the event queue
  + **Invoking Process Controller**
    - Send an EventTask or Business Object
      * Tells the Event Manager to run an event Detection Rule using an Adapter Def( if more info req)
  + **Sending the process controller an event task:**
    - EventTaskPArameters:
      * Accepts the system id, Bobj id,m Control and Bobj
      * Can add event Category to run for specified object
    - Time Based Trigger Event Detection
      * Process any event detections that has passed its next processing date
  + Event categories:
    - CreateSuspect
    - Retirement
    - ValuePAckage
  + **Event Detector Message Driven Bean:**
    - Receives the Event Task from the Process Controller
    - Invoke the process (EventTask) on the EventDetectorHelper
  + **EventTask:**
    - Holds EventTask Details
    - Uses Adapter(ADAPTERDEF) to retrieve Business object form MDM server
    - Retrieves previous events and calculates evemt horizon( after what amount of time should the event be rescheduled)
  + **EventDetector**
    - Invoked by EventDetectorHElper
    - EventDetection Rules are executed
    - Events found are passed back to EventDetector Helper
  + **How event Manager detects events?**
    - Uses the external rules component to configure business rules
    - Rules can be implemented in Java, or other pluggable rules engine
    - The business object fetched by the adapter is used as input to the business rule.
  + At the end of evemt detection:
    - If an event is detected:
      * Business rule creates a prnding EventObj and returns eventDetector
      * Event Persistence Module creates an EVENT record in database
      * Sends notification to a destination
    - If an event is NOT detected:
      * Resets the PROCESSACTION.NEXT\_PROCESS\_DT by adding the number days specified in the CDEVENTCAT.EVENT\_HORIZON
  + **ADDING NEWE EVENT DETECTION:**
    - **To add new event detection**
      * Define event category and definition
      * Write adapter to fetch business object
      * Write business rule to execute against the business object for that type of event
    - **BLACK BOX processing**:
      * Next process data reset
      * Selecting entities for time based event detection
      * Event persistence
      * Notification sending
* **4 Types of user interfaces in MDM-:**
  + **BUSINESS ADMINSITRATION UI:** 
    - Black box web application shipped with product and used to configure MDM.
    - Consists of following 5 modules:
      * **Rule Module**:
        + Manages ROV
        + Associates EOV with user groups
        + Manage group validation and element validation
      * Metadata module:
        + Create data associations
        + View and edit data associations
      * Code table module:
        + Management of Code table details
        + Error messages
        + Association of error messages with application components.
      * Security module:
        + Create user groups
        + Associate transactions with user groups
      * System module
        + Manage the extension framework
        + Manage the transaction audit information Log (TAIL)
  + **DATA STEWARDSHIP UI:**
    - White box web application and maintain Party data
    - Consists of 4 modules:
      * **Party Maintenance:  
         P**rovides the capability to:
        + Add or delete a person or organization
      * **Suspect Duplicate Processing:**
        + Search for suspect duplicates
        + Collapse multiple suspect duplicate party records
        + Manage the tasks for party record consolidation
        + Mark and unmark prospective duplicate suspects
        + Create two new parties from one party record
      * **Managing hierarchies**
      * **Managing Party Grouping:**
        + Search for parties and view a list of groupings associated with groups.
        + Create, maintain and view the grouping, grouping association, grouping values, and association roles
  + **PRODUCT UI:**
    - Maintain product information
    - Consists of:
      * Product maintenance
      * Hierarchy
  + **EVERGREEN CONSOLE**
    - Used for bulk standardization and duplication processing
* All the interfaces use **thin web browser client** :
  + UI Client🡪MDM UI Web application🡪 MDM Server🡪MDM DB
* Separate MDM UIs and Server
* **Clustering:**
  + **Physical:** Multiple Infosphere MDM instance running on different physical machines
  + **Logical:** Multiple Infosphere MDM instance running on same physical machine
* Two types of scaling:
  + Vertical
    - Adding capacity
    - Does not affect manageability, reliability
  + Horizontal
    - Adding servers
    - Increases manageability, reliability