Module 15 Physical MDM Architecture

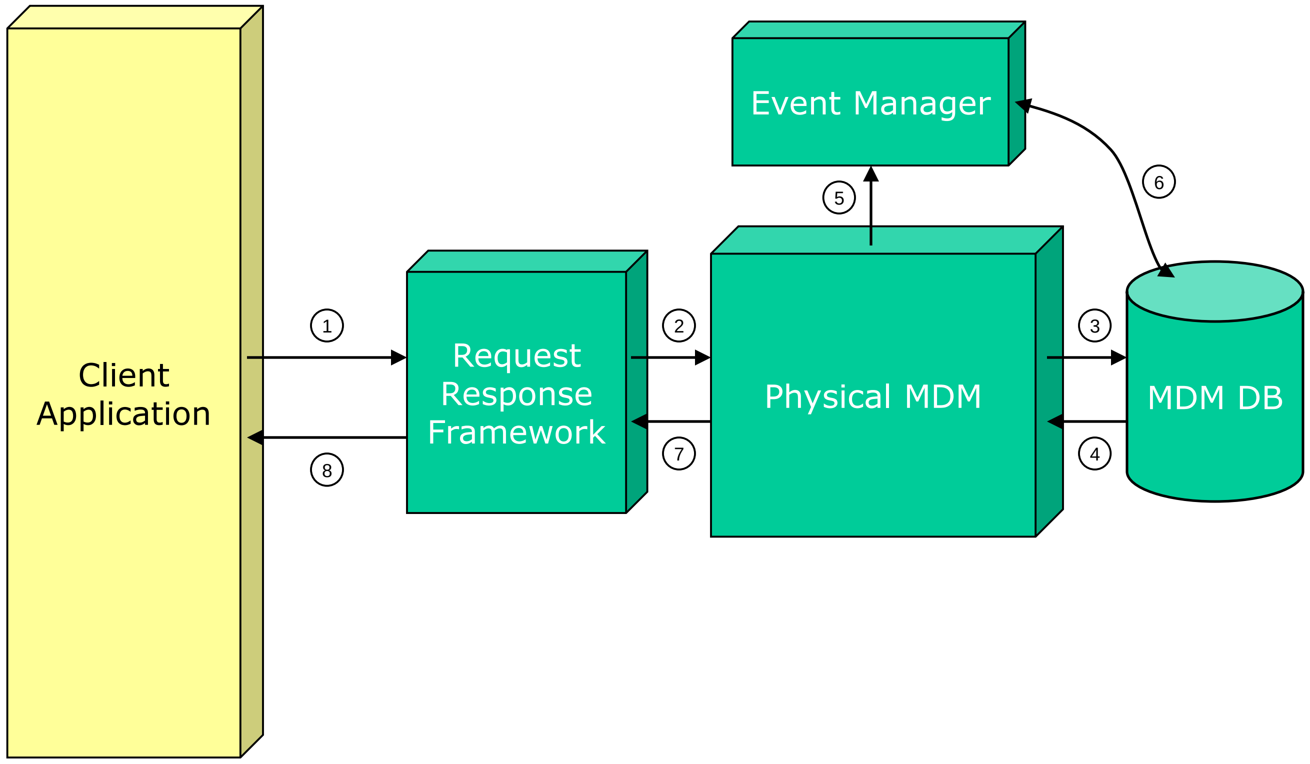
IBM InfoSphere Master Data Management

Module Objectives

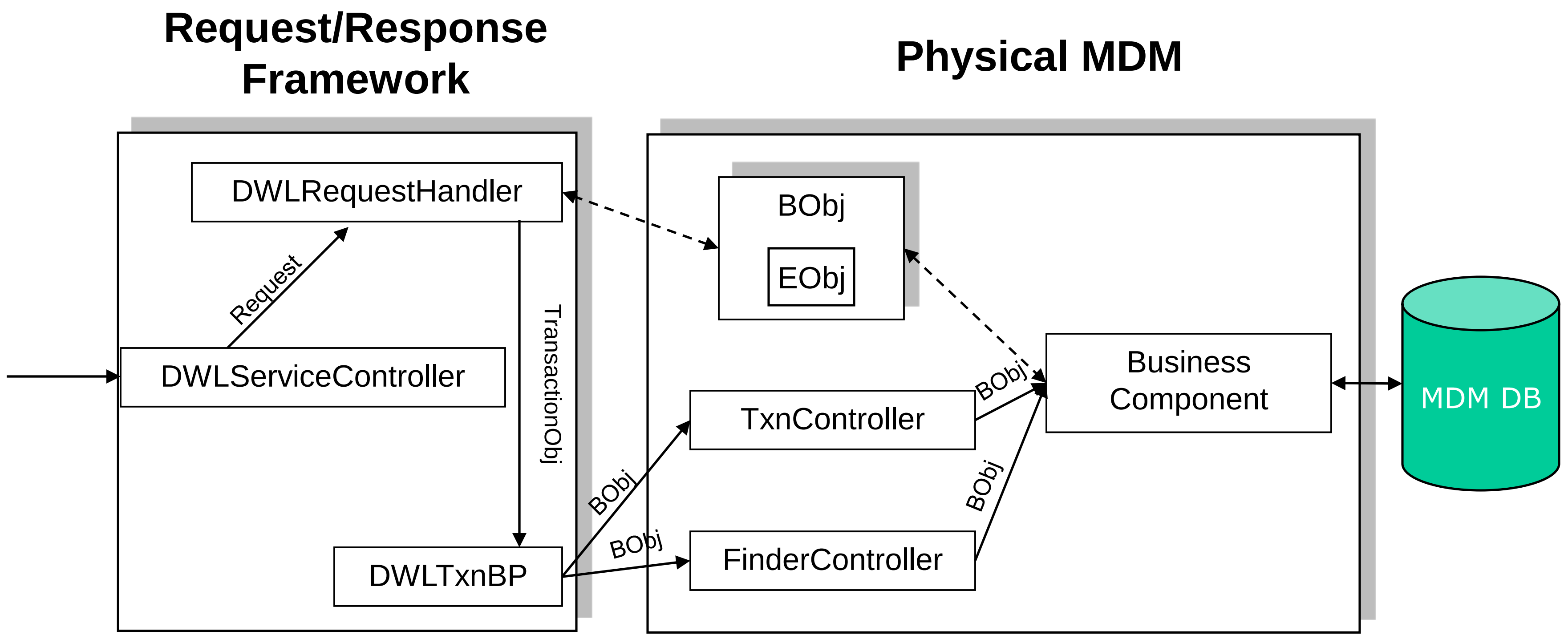
After completing this topic, you should be able to explain:

- the major components of the Physical MDM
- how a service is handled by the Request/Response Framework
- how a service is handled by the Physical MDM
- the major processes involved in the PrePost Framework

Physical MDM Architecture (100 foot view)



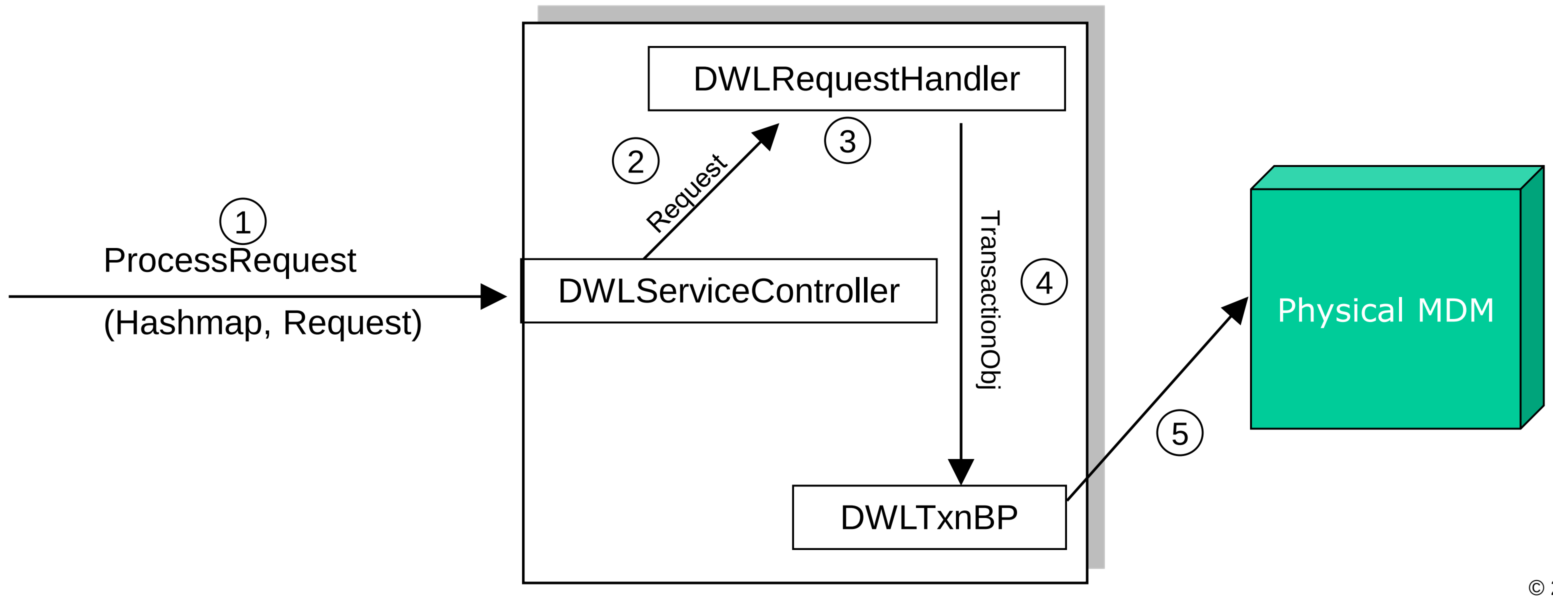
Physical MDM Architecture (10 foot view)



Request/Response Framework

1. Request is sent to the *DWLServiceController*.
2. *DWLServiceController* passes the Request to the *DWLRequestHandler*.
3. The *DWLRequestHandler* parses the request and creates a Transaction Object.
4. The Transaction Object is passed to the *DWLTxnBP*.
5. The *DWLTxnBP* call the appropriate service on MDM.

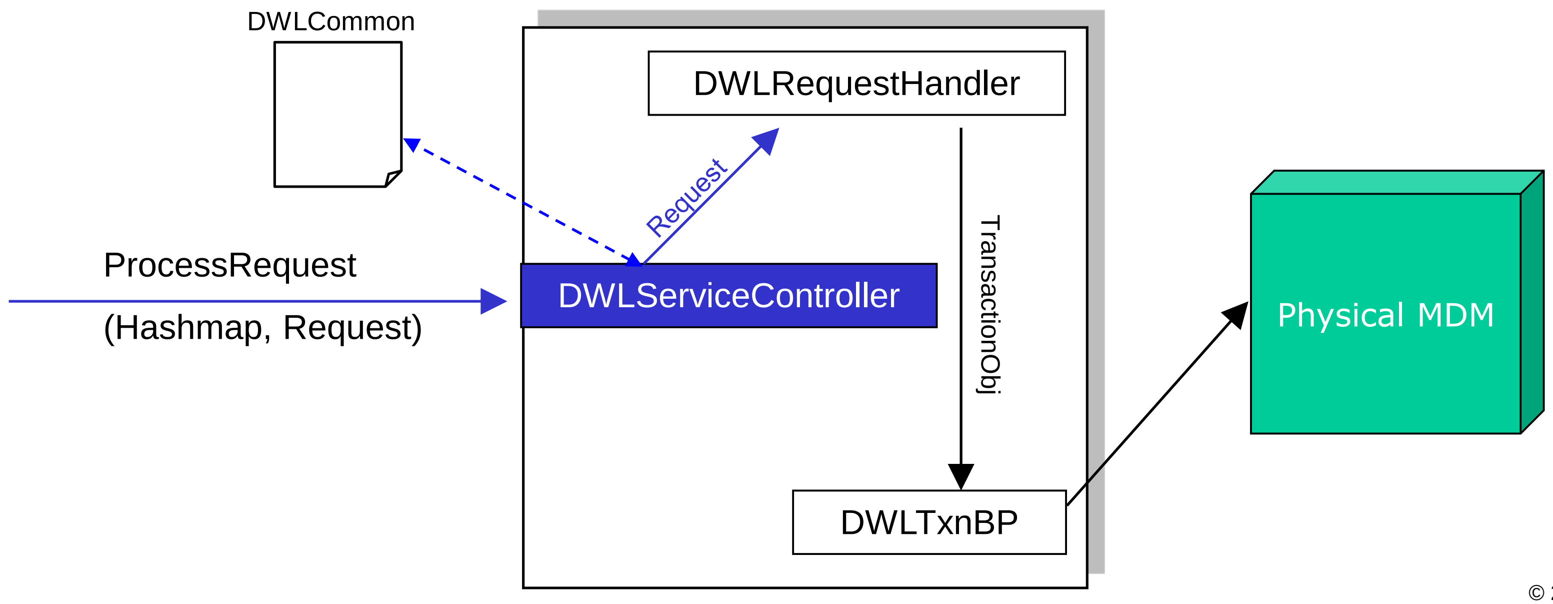
Request/Response Framework



DWLServiceController

1. Invoked using the ***processRequest(Hashmap txn context, Serializable request)***
   - The Hashmap contains (TargetApplication, Parser, Constructor, RequestType, ResponseType, OperationType, CompositeTxn, CompositeConstructor, CompositeParser, ASI_Request, ASI_Response)
2. Determines the Request Handler from the *DWLCommon.properties*.
   - RequestType.<ApplicationName>.<RequestType>
3. Starts the transaction.
4. Invokes the Request Handler (*DWLRequestHandler* by default).

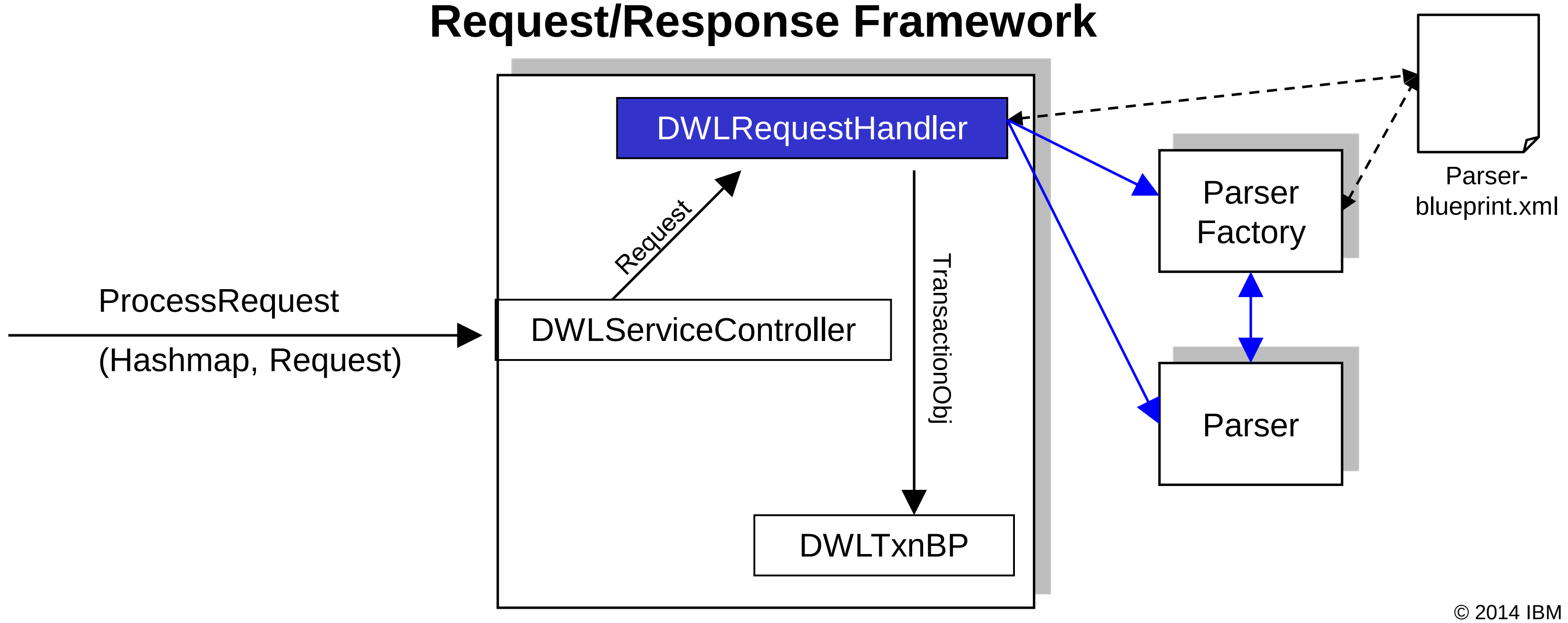
Request/Response Framework



DWLRequestHandler

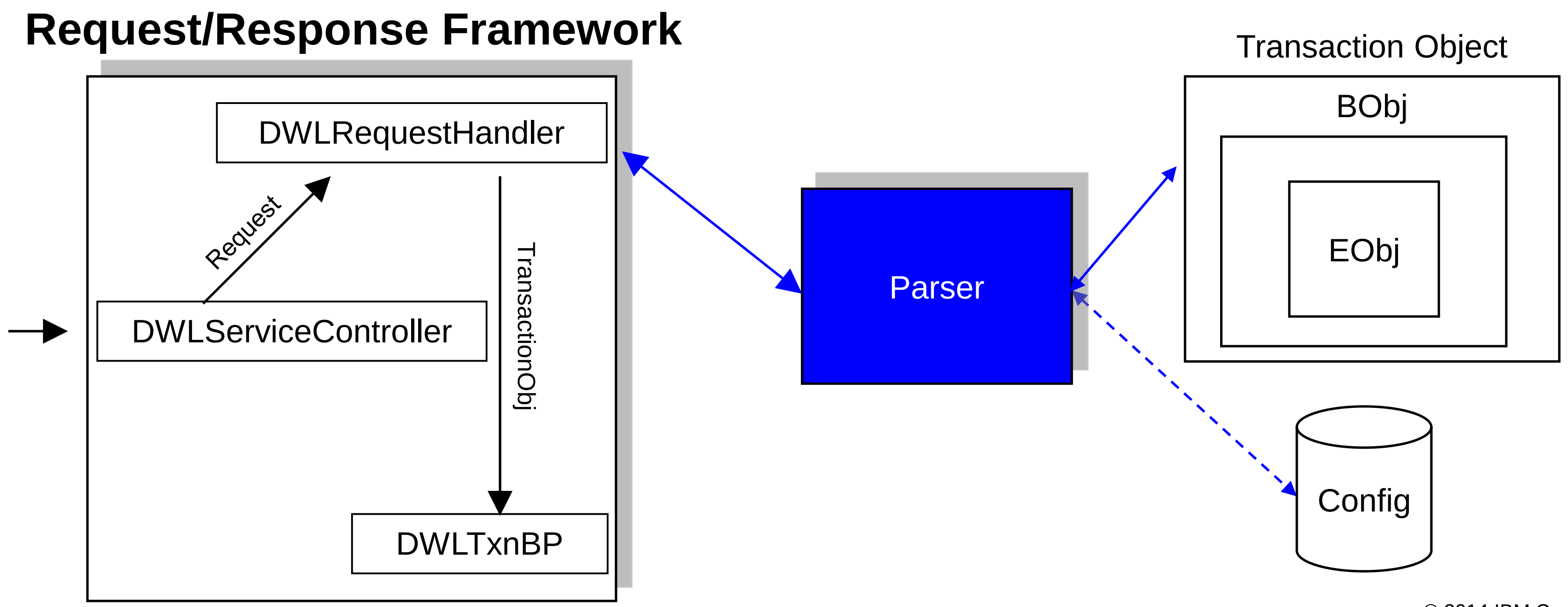
1. Invoked by the *DWLServiceController* and passed the Request
2. Acts as the main dispatcher for handling the request
3. Parser Factory is declared as an OSGi service with the names of the parsers in Parser-blueprint.xml
   - Default parser factory: com.ibm.mdm.base.requestHandler.ParserFactoryServiceImpl
4. Parser Factory determines the Parser based on txn context
   - Parser (eg. TCRMService)
5. Invokes the parseRequest on the Parser

Request/Response Framework



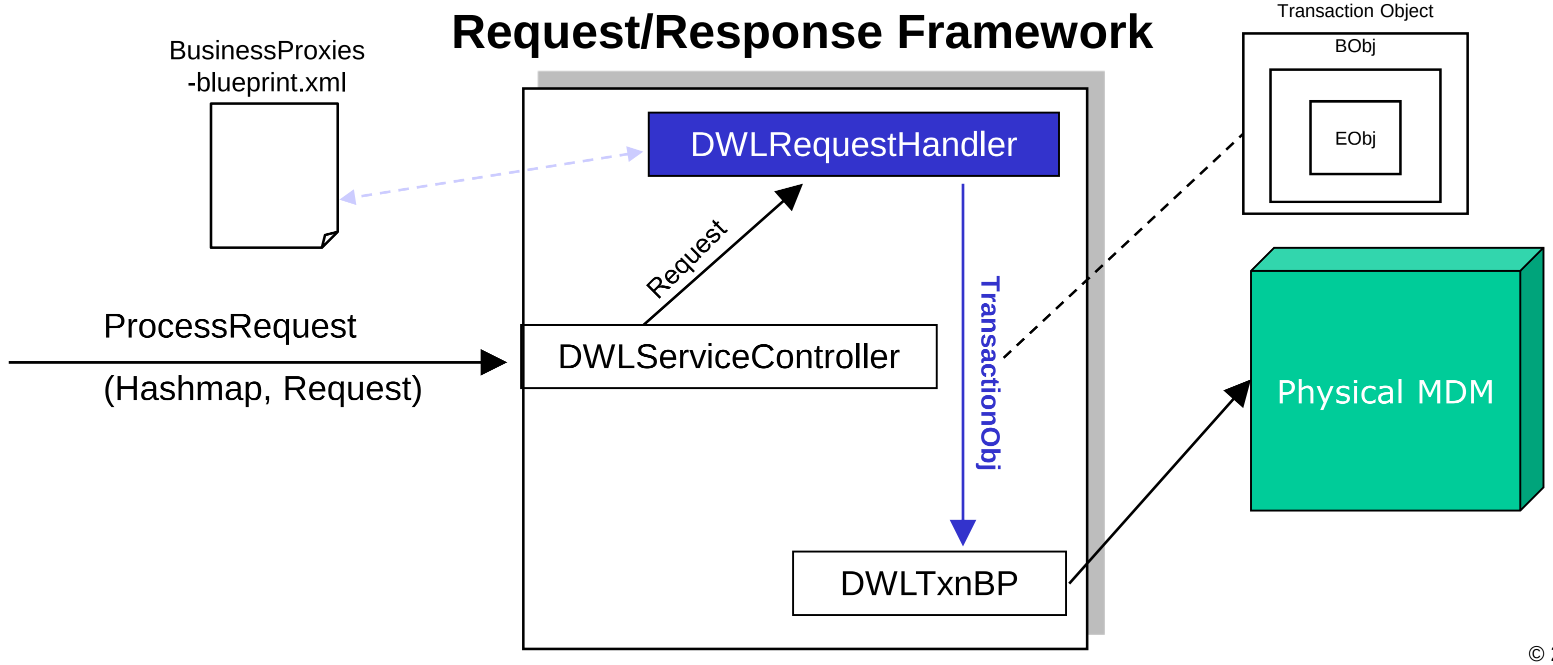
Parser

1. Constructs a Transaction based on the type of request (eg. addPerson).
   - A Transaction Object can be an InquiryTransaction (**I**), PersistentTransaction (**P**), or SearchTransaction (**S**)
   - The TX_OBJECT_TP column in the CDBUSINESSTXTP table defines the corresponding transaction object type
2. For Search and Persistent transaction, OSGi service, *BObj factory*, constructs the Business Object (BObj) passed in the request
3. The BObj constructs an Entity Object (EObj) that will hold the simple attributes of the BObj
4. The Transaction Object is passed back to the *DWLRequestHandler*



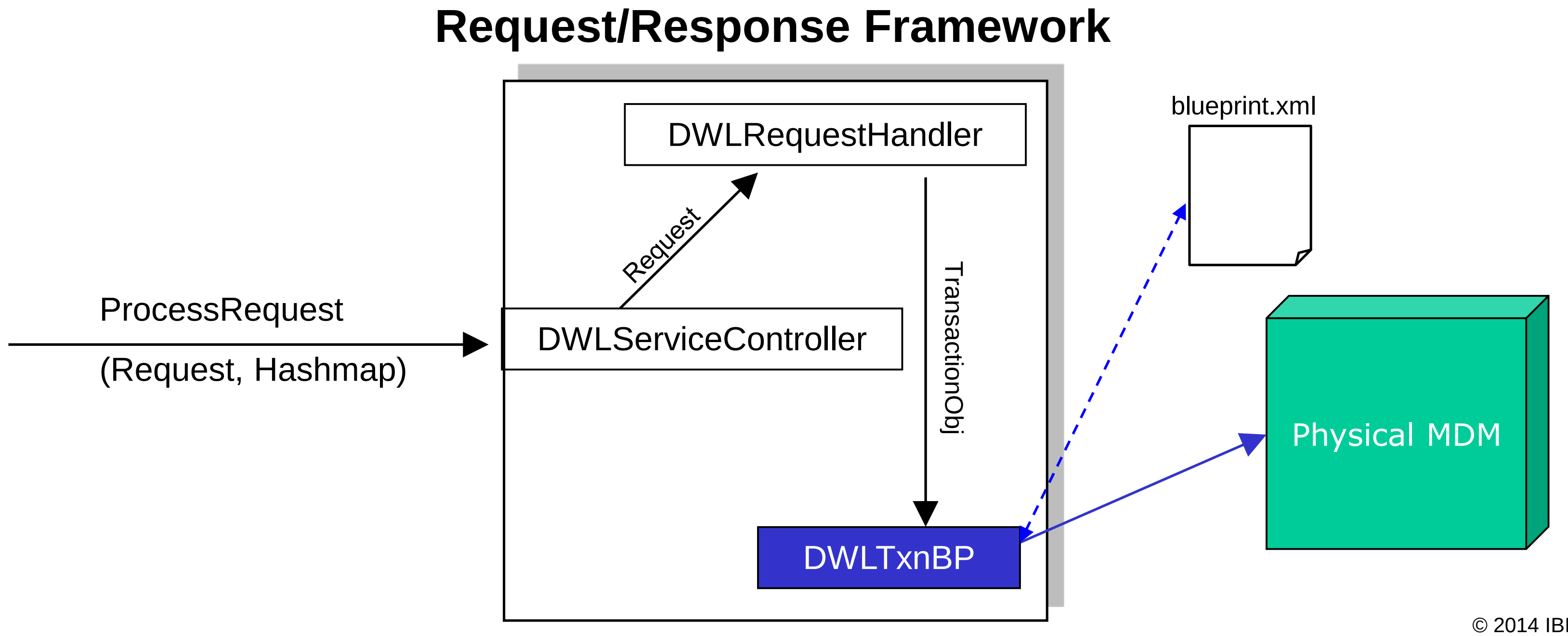
DWLRequestHandler

1. Receives the Transaction Object back from the Parser
2. Determines the business proxy using OSGi services that are business proxy factories based on transaction type
   - transaction.type: I, S, P
3. Invokes the business proxy, *DWLTxnBP,* and passes the TransactionObject



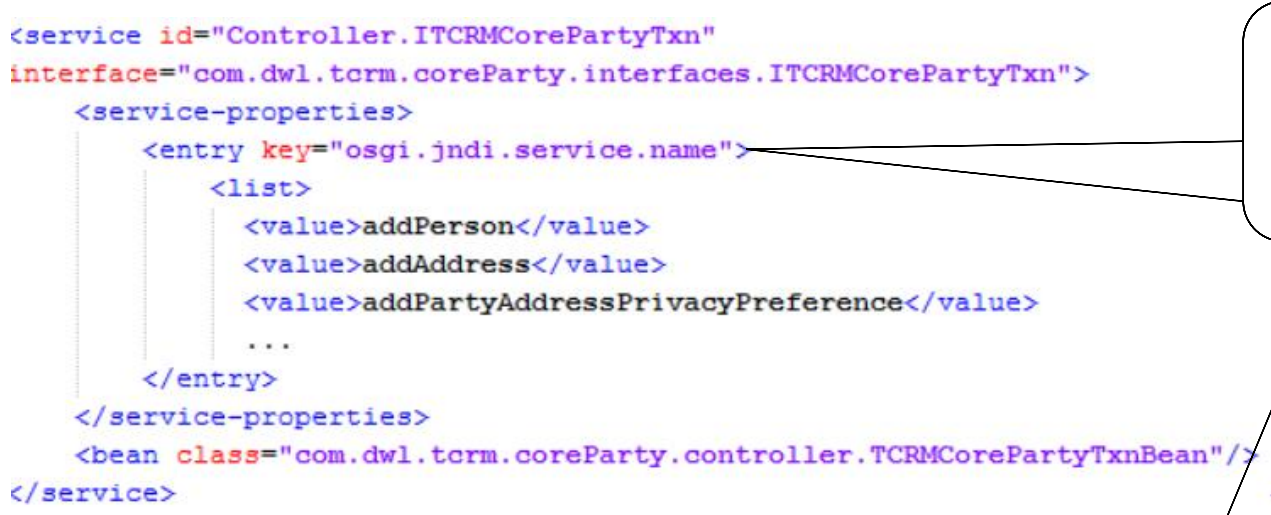
Business Proxy

1. Acts as a bridge between the Request/Response Framework and the Service Controller component
2. Provides the ability to compose new transactions by leveraging existing transactions (composite transaction)
3. *DWLTxnBP*
   - A default business proxy provided to interface with InfoSphere MDM
   - Delegates each incoming call to the appropriate InfoSphere MDM controller defined as OSGi service



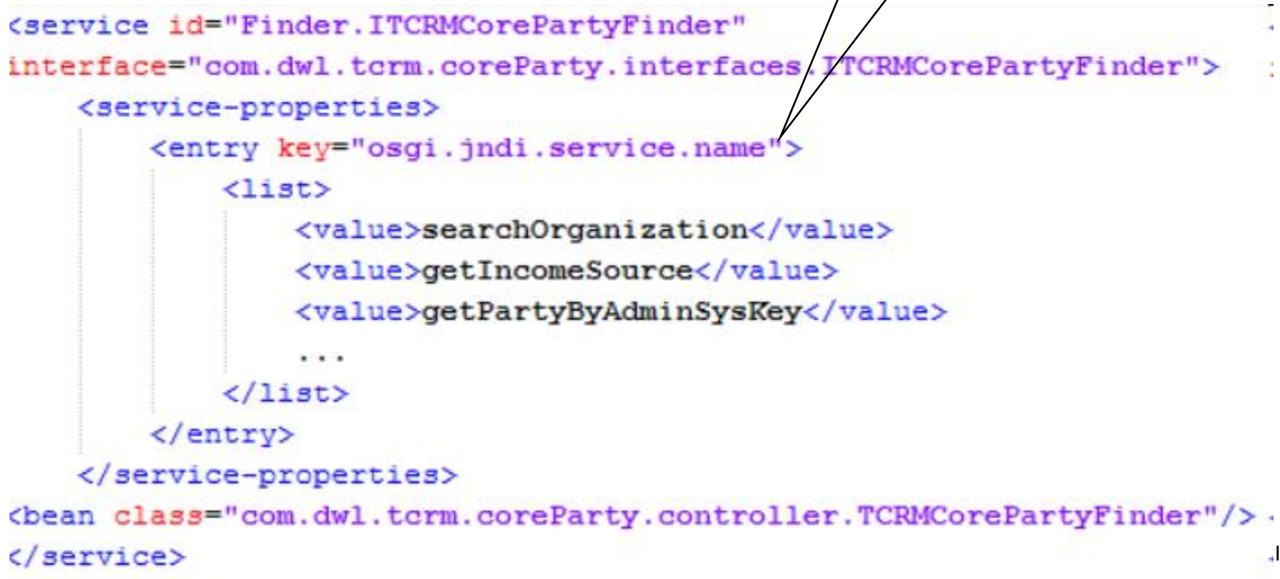
blueprint to find the right controller

- Controllers are now defined as OSGi services



```
<service id="Controller.ITCRMCorePartyTxn"
interface="com.dwl.tcrm.coreParty.interfaces.ITCRMCorePartyTxn">
    <service-properties>
        <entry key="osgi.jndi.service.name">
            <list>
                <value>addPerson</value>
                <value>addAddress</value>
                <value>addPartyAddressPrivacyPreference</value>
                ...
        </entry>
    </service-properties>
    <bean class="com.dwl.tcrm.coreParty.controller.TCRMCorePartyTxnBean"/>
</service>
```

the supported transactions are listed as service properties attached to a key called key="osgi.jndi.service.name

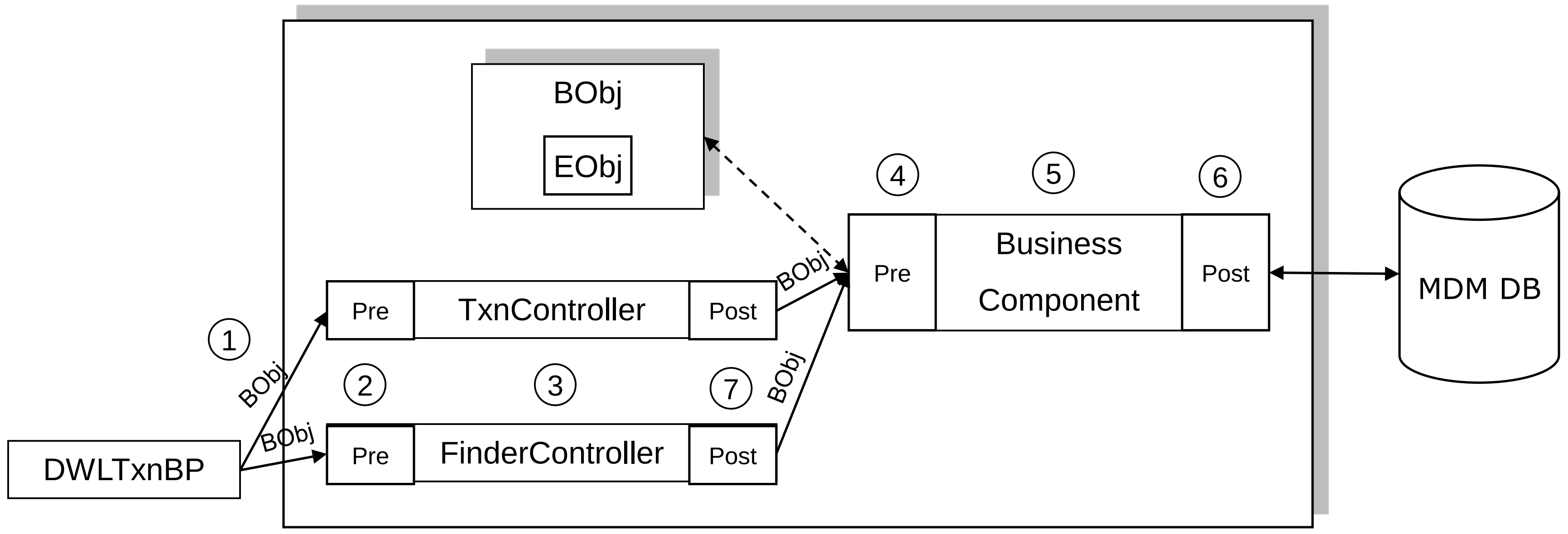


```
<service id="Finder.ITCRMCorePartyFinder"
interface="com.dwl.tcrm.coreParty.interfaces.ITCRMCorePartyFinder">
    <service-properties>
        <entry key="osgi.jndi.service.name">
            <list>
                <value>searchOrganization</value>
                <value>getIncomeSource</value>
                <value>getPartyByAdminSysKey</value>
                ...
            </list>
        </entry>
    </service-properties>
<bean class="com.dwl.tcrm.coreParty.controller.TCRMCorePartyFinder"/>
</service>
```

Core Physical MDM

1. The *DWLTxnBP* Invokes the appropriate Controller
   - Persistent transaction will go to Txn Controllers
   - Non-persistent transactions will go to Finder Controllers
2. The Controller delegates to the work to the appropriate Business Components.
3. The Business Component access the EObj from the BObj.
4. The Request is then Persisted to the database or retrieved from the database using PureQuery.

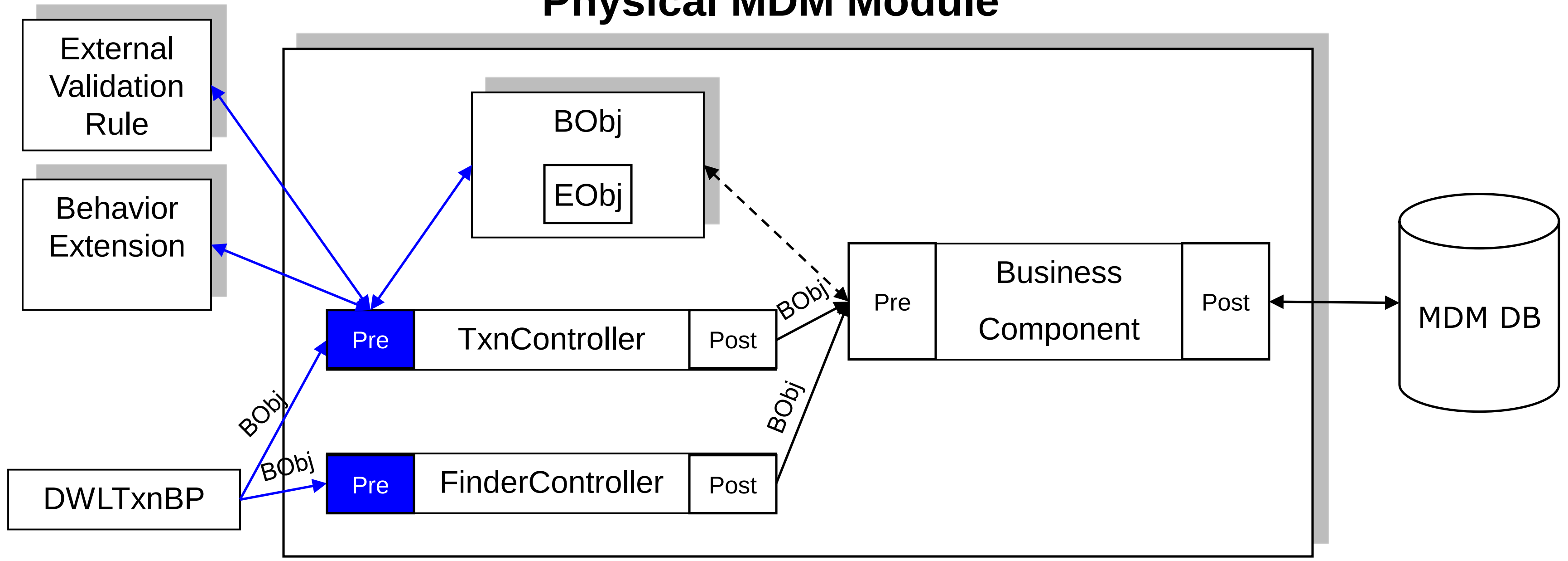
Physical MDM Module



Pre-Controller

1. Prior to running the transaction in the Controller, the pre-execute runs the following:
   - **External Validation** – V_Group_Val, V_Group_Param, V_Element_Val, V_Element_Param, V_Function
   - **Internal Validation** – Level 1 (validateAdd, validateUpdate, validateView methods in BObj)
   - **Pre-Transaction Behavior Extensions** – ExtensionSet table

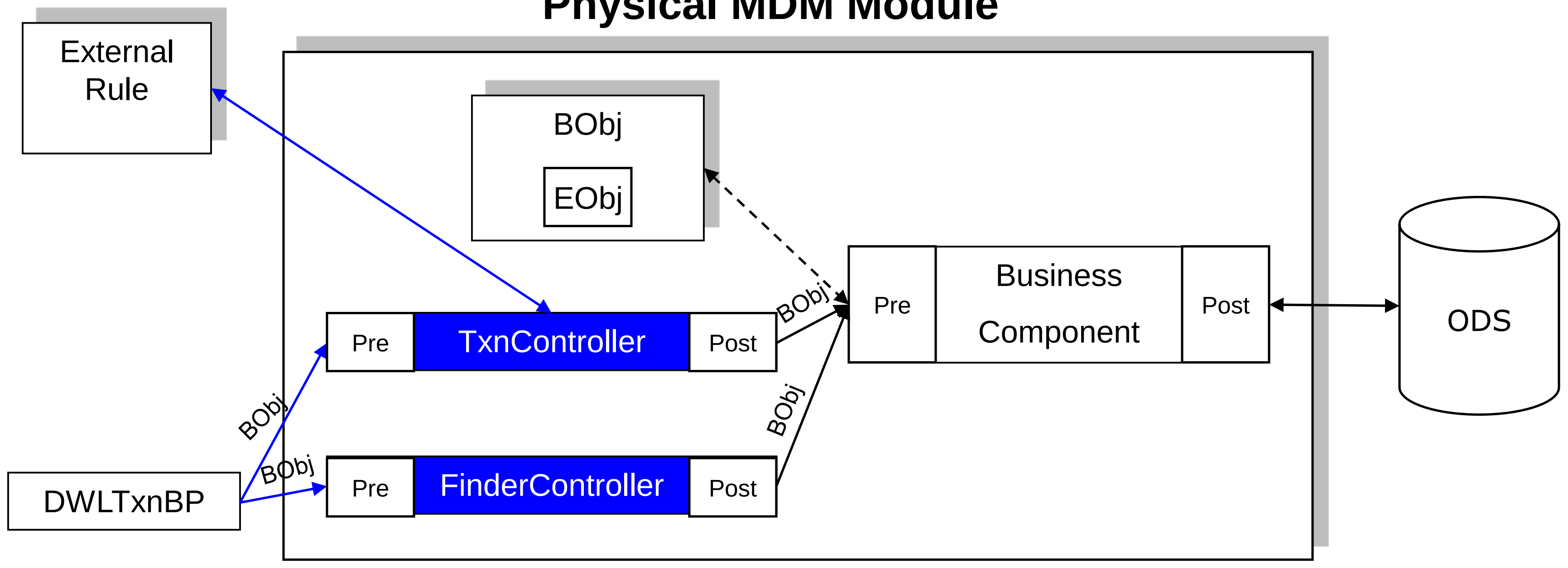
Physical MDM Module



Controller

1. During the execution of the service at the Controller level, External Rules might be executed.
2. External Rules exposes logic to client that can be modified (eg. Searching for Duplicate Records).
3. The Controller Delegates to the work to the appropriate Business Components.

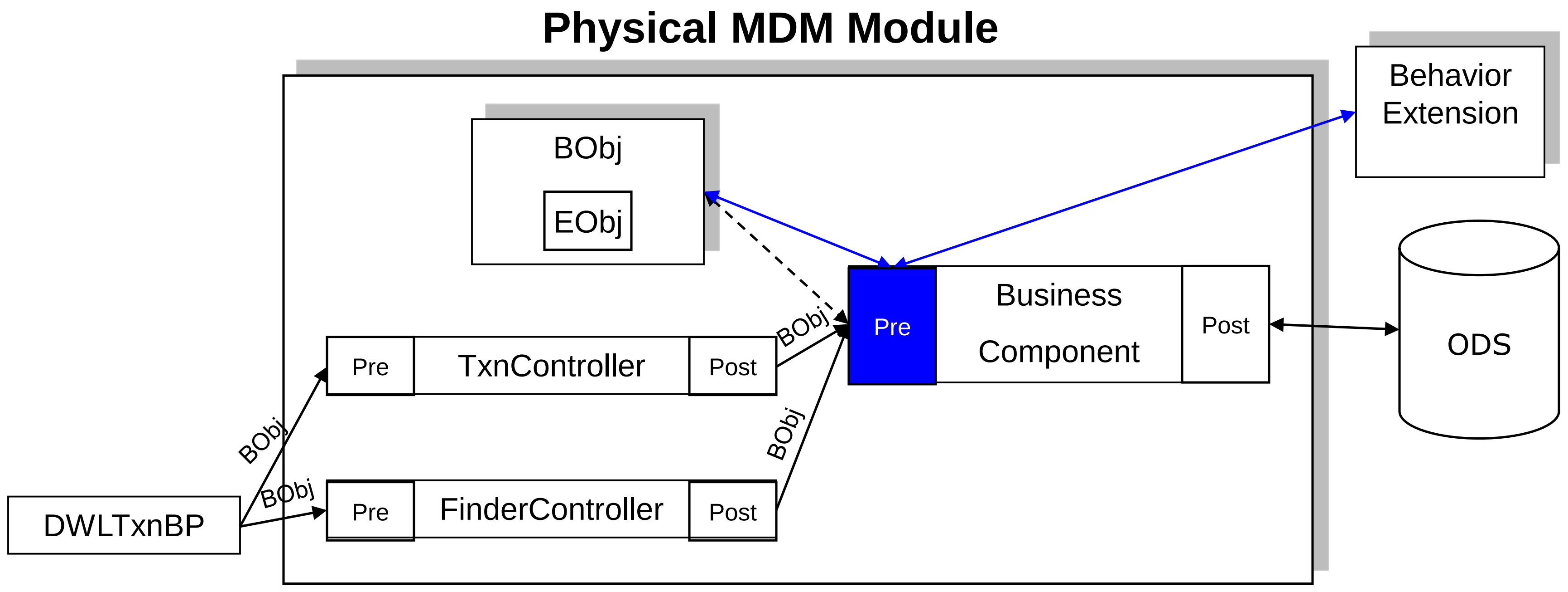
Physical MDM Module



Pre-Component

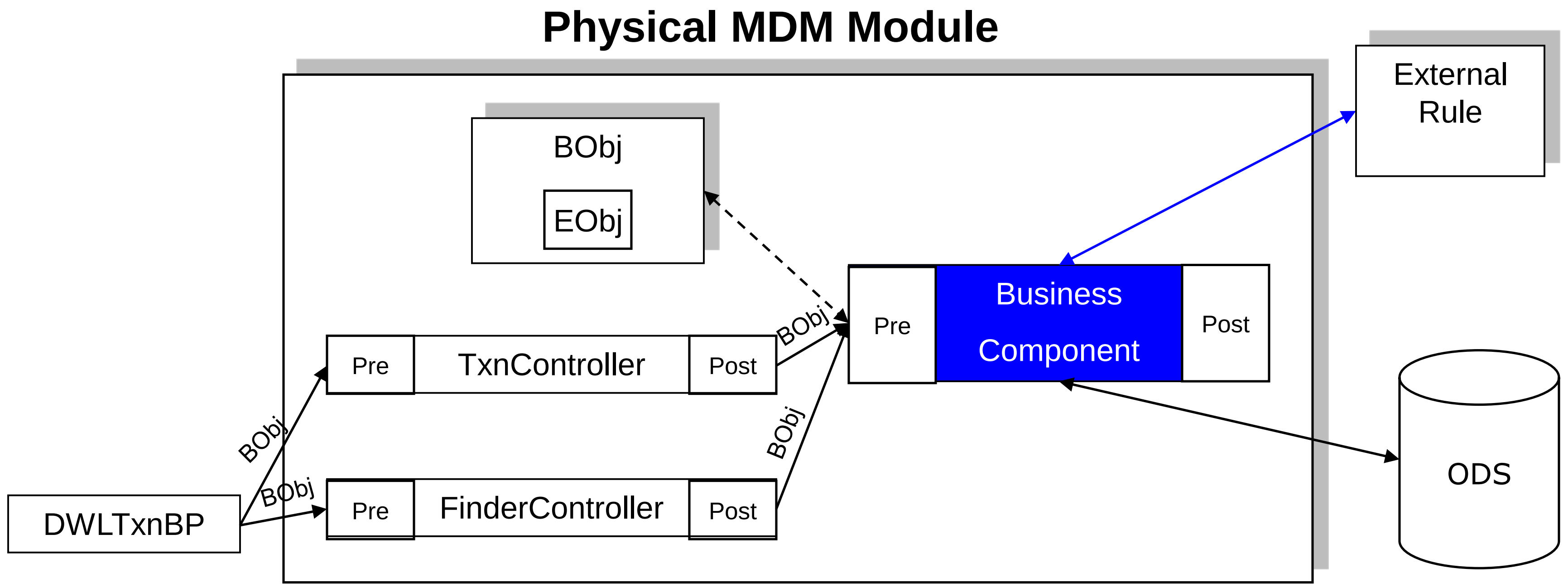
1. Prior to running the transaction in the Component, the pre-execute runs the following:

- **Internal Validation** – Level 2 (validateAdd, validateUpdate, validateGet methods in BObj)
- **Pre-Action Behavior Extensions** – ExtensionSet table



Business Component

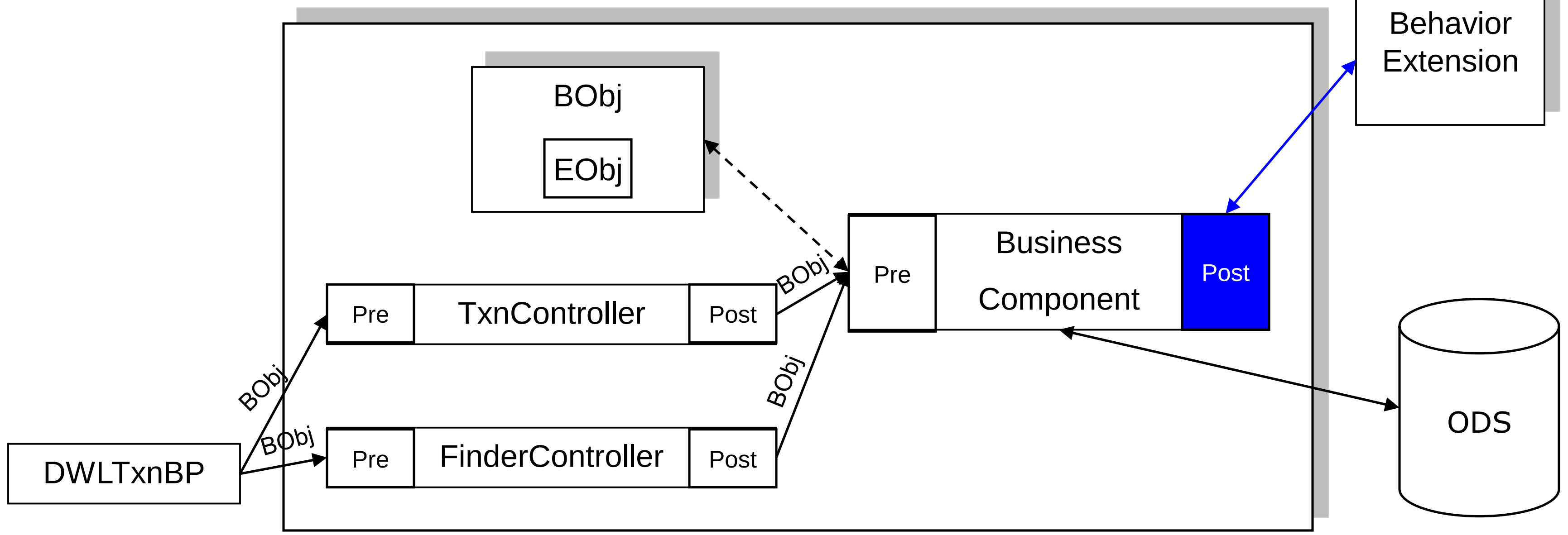
1. Will retrieve the EObj from the BObj for non-inquiry transactions (transactions that use BObjs).
2. Will persist or retrieve the data from the database using PureQuery.
3. For inquiry transaction will build the new BObj to pass back to the calling client.
4. External Rules exposes logic to client that can be modified (for example, Standardization).



Post-Component

1. After running the transaction in the Component, the post-execute runs the following:
   - **Post-Action Behavior Extensions** – ExtensionSet table
   - **Transaction Audit Information Log** (TAIL) collection

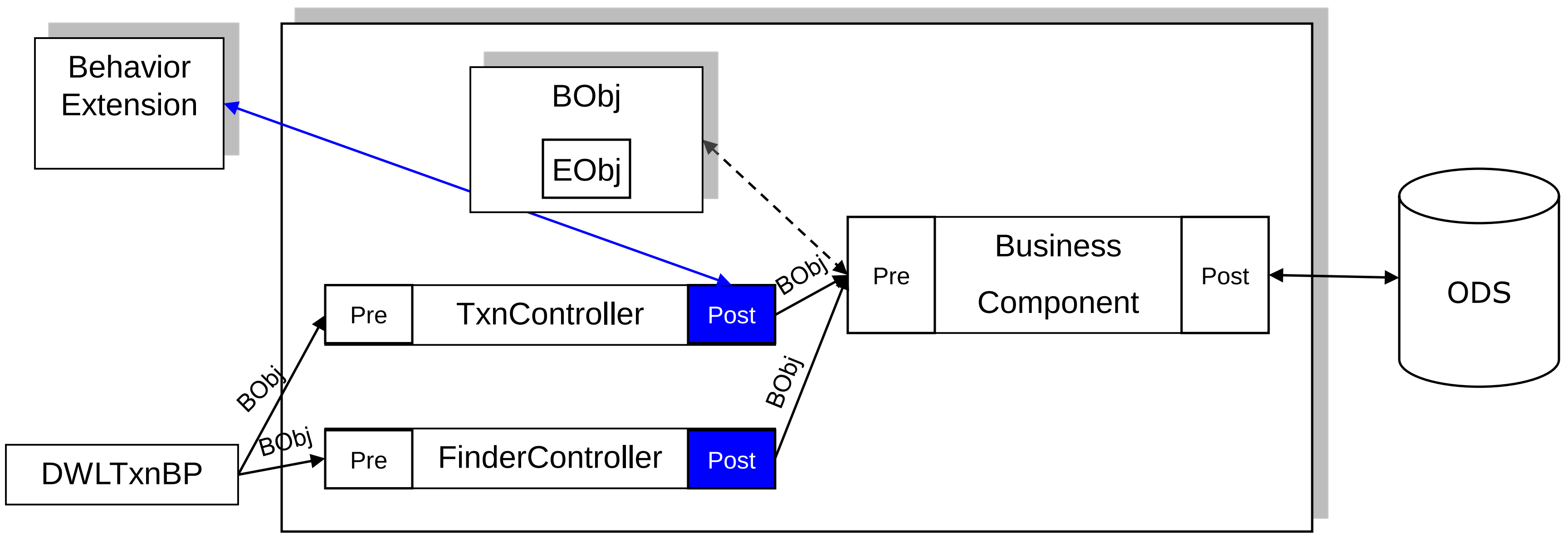
Physical MDM Module



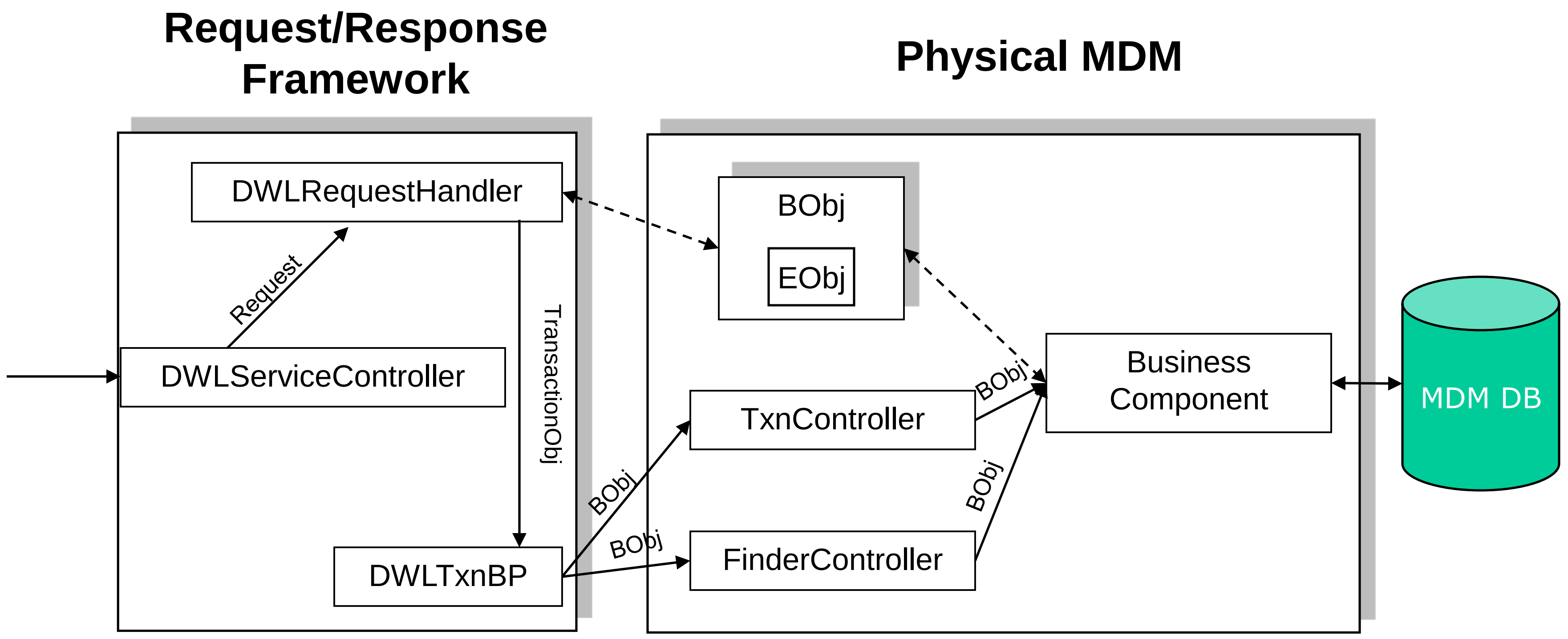
Post-Controller

1. After running the transaction in the Post-Componet, the post-execute runs the following:
   - **Post-Transaction Behavior Extensions** – ExtensionSet table
   - **Transaction Audit Information Log** (TAIL) collection

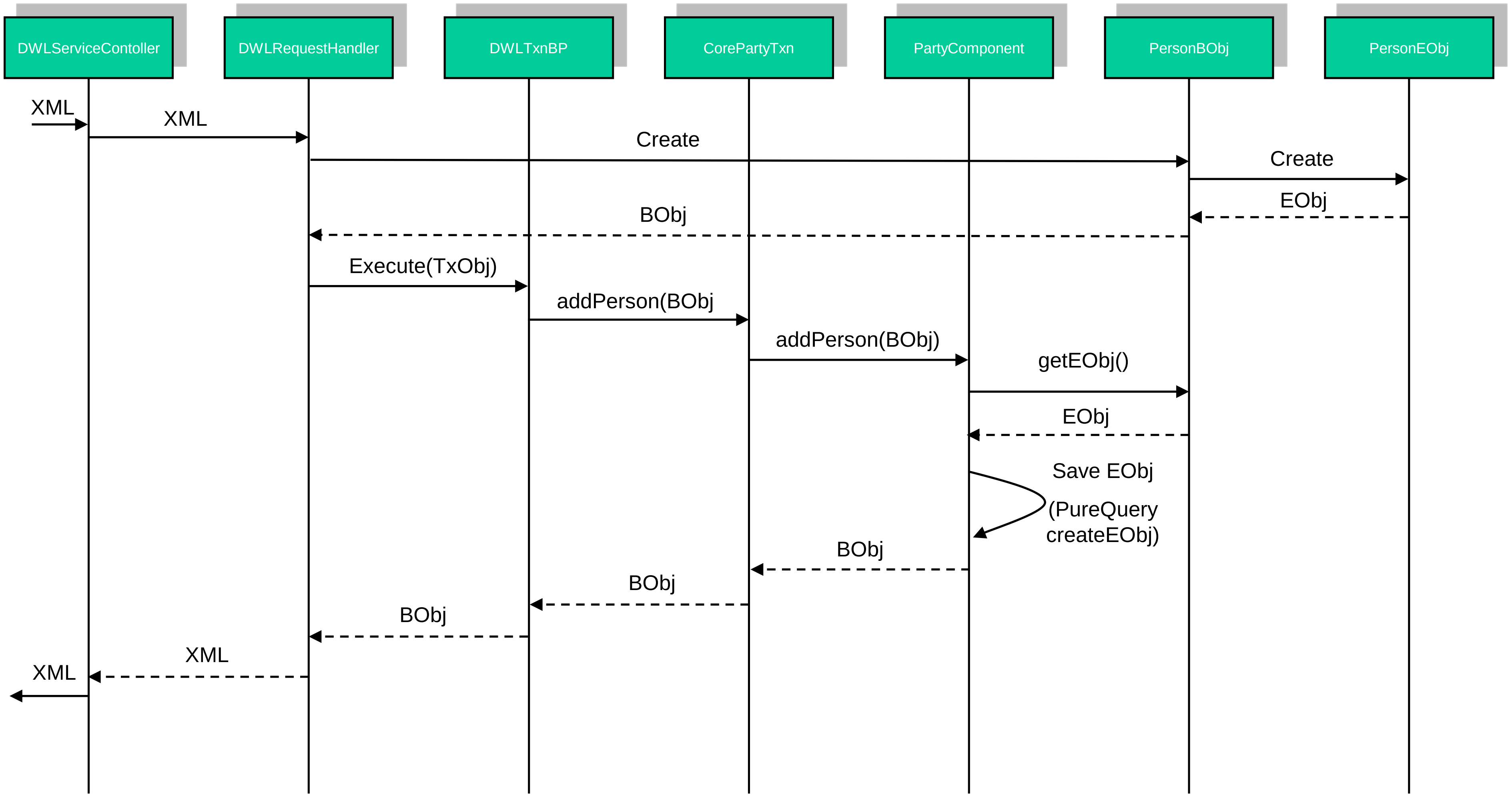
Physical MDM Module



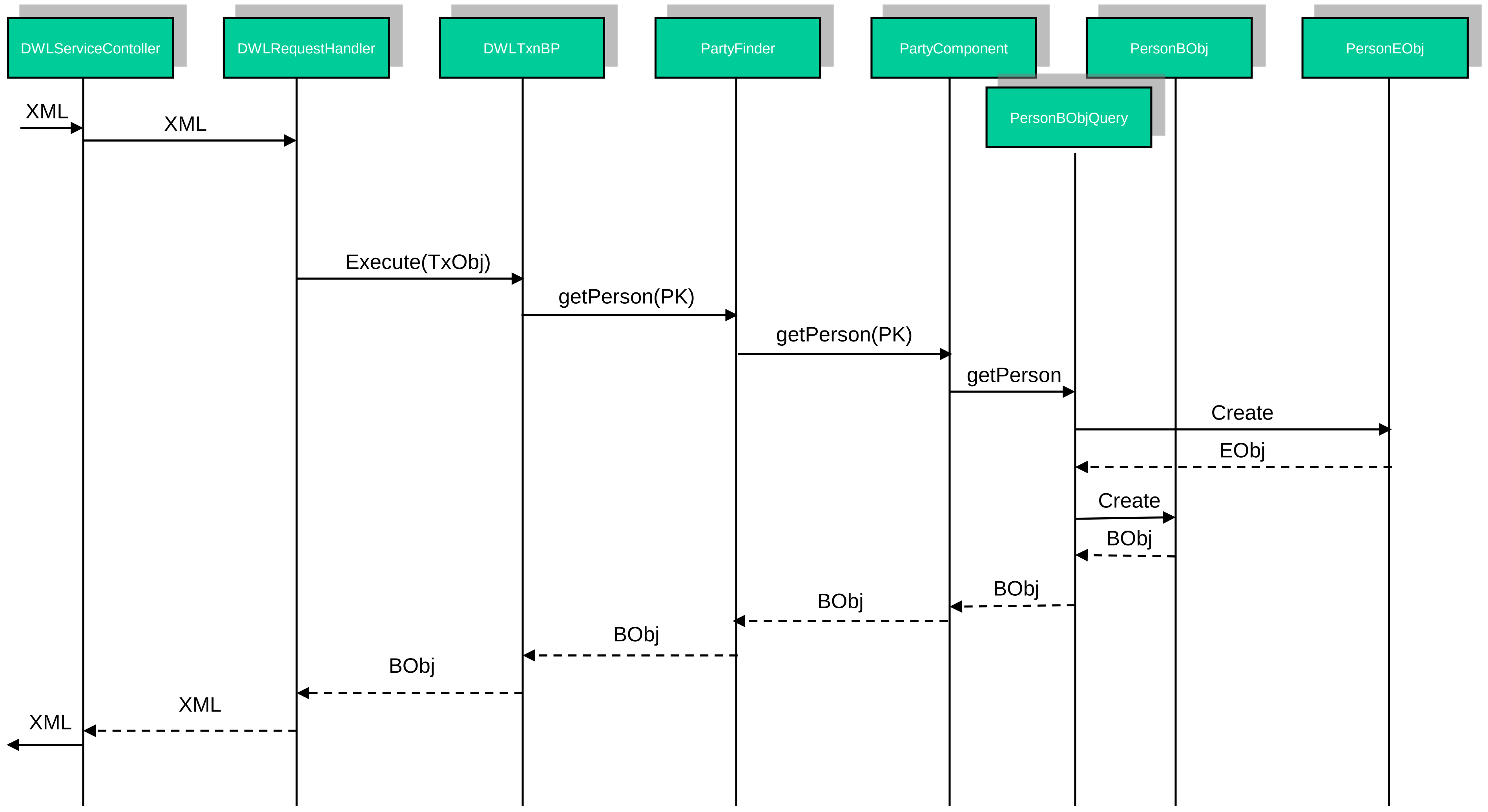
Physical MDM Architecture (10 foot view)



The ‘AddPerson’ Sequence



The ‘GetPerson’ Sequence


DWLServiceContoller
DWLRequestHandler
DWLTxnBP
PartyFinder
PartyComponent
PersonBObj
PersonEObj
PersonBObjQuery
XML
XML
Execute(TxObj)
getPerson(PK)
getPerson(PK)
getPerson
Create
EObj
Create
BObj
BObj
BObj
BObj
BObj
XML
XML