# COMMONWEALTH OF PENNSYLVANIA
# DEPARTMENT OF HUMAN SERVICES

# INFORMATION TECHNOLOGY STANDARD

| Name Of Standard: **Cognos Model/Package Development** | Number: **STD-EKMS008** |
|---|---|
| Domain: **Knowledge Management** | Category: **Data Warehouse/Business Intelligence** |
| Date Issued: **08/18/2008** | Issued By Direction Of: |
| Date Revised: **02/17/2016** | Clifton Van Scyoc, Dir of Division of Technology Engineering |

## Abstract:

The purpose of this standard is to establish enterprise-wide standards and guidance for the creation of Framework Manager projects.

Cognos BI provides a variety of reporting, analysis, dashboard, and scorecard capabilities to provide the right amount of detail to the report consumer. The models and packages created by the developers via the Cognos BI Framework Manager tool are the foundation upon which all Cognos BI reporting rests. The Framework Manager model is the basis for SQL generation of business intelligence and reporting code created when any attribute is selected for a specific report.

## General:

This standard applies to all Department of Human Services Cognos developers, both Commonwealth and contractor. This standard outlines production implementation procedures for all Framework Manager projects created, regardless of report database source or report audience.

## Definitions:

**Subject Area:** The group of related data items in the Data Warehouse that are generated by, and used in, one or more electronic systems or business processes. The name for a subject area should be approved by the end user prior to starting a model. Example: Medical Assistance Claims, Application Processing, HCSIS, Pelican

**Model:** A Framework Manager implementation of a single subject area in the Data Warehouse. A model should present the technical elements of the Data Warehouse (tables, fields, relationships, etc) in a way that supports reporting/data analysis by non-technical users.

**Namespace:** EKMS uses the following terminology to distinguish namespaces with specific roles and meanings from standard namespace objects.

- **Model Namespace:** The topmost namespace in a CPF file.
- **Layer Namespace:** Defines a "layer" in the model. Each layer in the model…
    - Defines the types of framework manager objects found within it
    - Defines the actions the modeler must perform.
    - Has a specific purpose.
    - Is designed to ease maintenance and encourage reuse of the model.
- **Relationship Namespace:** A type of namespace found in the 3rd layer of the model. A relationship namespace contains only shortcut objects.
- **Presentation Namespace:** A type of namespace found in the 4th layer of the model.

**Folder:** EKMS uses the following terminology to distinguish folders with specific roles and meanings from standard folder objects.

- **Database Schema Folder:** A type of folder containing Data Source Query Subjects and Model Query Subjects in the 1st and 2nd layer of the model. This folder logically groups and identifies query subjects with the underlying database schema they are sourced from.
- **Relationship Folder:** A type of folder found in the 3rd layer of the model within a Relationship Namespace. Relationship Folders contain the shortcut objects that define the star-schema/snowflake groupings used by a Model Query subject in the 4th layer of the model.

**Project:** A single CPF file which contains **one** model and one or more packages. (A single project may link to multiple models, but each model resides in its own CPF file.)

## Standard:

### Projects

This standard is continuously updated to remain current with industry standards and software updates. Before any brand new Framework Manager model development is begun, EKMS should be consulted for any updates to this standard.

BIS must approve the creation of new projects. Approval will only be given when a new project meets the following criteria:

- The subject area(s) to be modeled in the new project are not present in an existing project. If the subject area(s) to be modeled already exist, there must be a valid business justification for the duplication of effort. In the majority of cases preference should be given to updating/correcting an existing project.
- The subject area(s) to be modeled cannot be added to an existing project due to technical reasons. (I.e. the resulting project size would make maintenance difficult; the resulting project size would negatively affect reporting performance; the design of the existing project, etc...)

All models utilized by production reports must be provided to BIS as part of a BI deployment release so they can be included in the DHS Framework Manager Model Library.

All new projects will be created following these standards which will allow linking to larger projects.

1. The project (CPF) filename shall be in the format "*Subject Area Name* Project." Example: Expedited Medical Encounters Project. CPF
2. The Project Design Language shall be 'English' only.
3. The Model Namespace shall be named "*Subject Area Name* Model." Example: Expedited Medical Encounters Model

4. Each project should contain one model and one or more packages. (A single project may link to multiple models, but each model resides in its own CPF file.)

## Governors

1. The following governor's should be set for the project (Project Menu > Edit Governors…)
    a. SQL Join Syntax = Explicit
    b. Use WITH clause when generating SQL: Checked

## Data Sources

2. After starting a new Framework Manager project and importing tables for the first time, the Data source name will need to be manually changed to meet the naming conventions supplied in this document.
3. To add a new table to an existing Data source:
    a. Right click Database Layer
    b. Run MetaData Wizard
    c. Choose Data Source
    d. Select correct Data Source
    e. Select correct schema
    f. Select needed table(s)
    g. Uncheck "Use Primary and Foreign Keys"
4. In the **Project Function List** (menu), **Define Quality of Service** and modify the **Vendor Specific Functions** so that only those of the database used by the project are available to the user.

## Models

In Framework Manager, a model provides a physical and business view of the data from one or more data sources. It describes objects and their relationships to be used in the BI environment.

This standard takes into account that the same underlying data may need to be joined and presented in multiple ways depending upon end user requirements.

<u>Dimensional Modeling and Star Schema:</u>

Per Data Warehouse best-practice, all models will follow dimensional modeling guidelines and use fact/dimension tables organized into star-schemas. Proper dimensional design will take place in the underlying database prior to the model's development. Data Warehouse surrogate keys will be used to join fact and dimension tables; operational system keys and/or business keys are not acceptable as join keys.

All new projects to be used in Cognos must be developed using dimensional models. The only exception is existing models and modifications to existing tables and EDW structures.

Models should not be based on normalized tables or contain "snowflake" or "outrigger" tables. The creation of Model Query Subjects (MQS) to simulate proper dimensional design of non-dimensional tables, or flatten "snowflaked" dimensions, is prohibited.

Failure to follow dimensional modeling best practice will result in models that produce unpredictable SQL queries, generate incorrect summarizations, are less intuitive to use, and prevent Cognos from running in the higher performance Dynamic Query Mode (DQM).

<u>Model Layers:</u>

Cognos BI Framework Manager provides the capability to create multiple layers within a model. This is similar to the physical and logical layers of an entity relationship diagram. The resulting models will consist of four layers as outlined and described below:

| | Objects In Layer | Responsibilities of the Modeler | Notes |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **LAYER 1** | • Data Source Query Subjects (DSQS)<br>• Database Schema Folders | • Create Database Schema Folder(s).<br>• Create DSQS.<br>• Set DSQS properties<br>• For each query item in a DSQS set the following properties:<br>o Usage<br>o Aggregate properties<br>o Format<br>o Description | • One DSQS per database table.<br>• DSQS are named after the database table they reference.<br>• Query items within each DSQS retain their database names.<br>• There must be at least one Database Schema Folder in this layer.<br>• Use multiple Database Schema Folders as required. |
| | | | |
| **LAYER 2** | • Model Query Subjects (MQS)<br>• Database Schema Folders | • Create Database Schema Folder(s).<br>• Create MQS.<br>• Create embedded filters on MQS as necessary.<br>• Assign data-level security to MQS as necessary.<br>• Set Determinants as required.<br>• Ensure that properties for Query Subjects and Query Items in lower layers of the model are inherited. | • One MQS per DSQS. (More than one MQS per DSQS if additional MQS are required to support different embedded filters or data-level security.)<br>• See naming conventions for MQS names.<br>• Query items maintain database names.<br>• There must be at least one Database Schema Folder in this layer.<br>• Use multiple Database Schema Folders as required. |
| | | | |
| **LAYER 3** | • Relationship Namespace(s)<br>• Relationship Folders<br>• Alias Shortcuts to MQS in Layer 2<br>• Reference Shortcuts to MQS in Layer 2<br>• Relationships (joins) between shortcuts. | • Create Relationship Namespace(s).<br>• Create Relationship Folder(s).<br>• Create Shortcut objects.<br>• Create Relationships (joins) between Shortcut objects. | • There must be at least one Relationship Namespace in this layer.<br>• Use multiple Relationship Namespaces as required.<br>• There must be at least one Relationship Folder in this layer.<br>• Use multiple Relationship Folders as required. |
| | | | |
| **LAYER 4** | • Presentation Namespace(s)<br>• Namespaces<br>• Model Query Subjects (MQS)<br>• Stand-Alone Calculations<br>• Stand Alone Filters | • Create Presentation Namespace(s)<br>• Create additional Namespaces within each Presentation Namespace to support user requirements<br>• Create MQS to support user requirements<br>• Assign business names to Query Items<br>• Define data object security on MQS<br>• Create Calculations and Stand Alone Filters as necessary to support business needs<br>• Assign object-level security to MQS and query items as necessary.<br>• Ensure that properties for Query Subjects and Query Items in lower layers of the model are inherited. | • One Presentation Namespace per presentation of the data.<br>• The layout of namespaces and MQS within each Presentation Namespace should be approved by the user.<br>• The names for all objects in this layer should be approved by the user.<br>• MQS are sourced from shortcuts in Layer 3. |

<u>Alphabetical Arrangement of Framework Manager Objects</u>

- All objects should be arranged alphabetically (top to bottom when viewed in the "Project Viewer" pane within Framework Manager) within their enclosing objects. That is:
    - Namespaces within Layers
    - Namespaces within other Namespaces
    - Folders within a Namespace.
    - Model Query Subjects/Shortcuts within a Namespace or Folder.

- Query Items within a Query Subject.

LAYER 1: Database Layer

1. To ease maintenance the Database Layer will be composed entirely of Data Source Query Subject (DSQS) objects and folders. Each table to be used in the model will be represented by one DSQS. There is to be no duplication of DSQS for the same table in this layer.
2. Within the Database Layer a folder should be created for each database schema used in the model.
3. The database schema folder will be named the same as the schema it represents.
4. All DSQS should reside in the same database schema folder as the schema they are sourced from.
5. DSQS should be SELECT * from the table rather than selecting specific fields.
6. The "Description" property of all DSQS should be set to contain a description of that DSQS and the table it is based on. The description should convey a rich understanding of the table and its data and should be written using the following guidelines:
    a. This description should be understandable to a layman and assume no prior knowledge of the data contained in the table or its associated business processes. This description may be read by end-users and future model maintainers who are not subject matter experts, and have no experience with, the table or its data.
    b. The table's granularity should be described (for fact tables) as well as any hierarchies present in the table (for dimension tables). Any partitions and indexes on the table should be listed along with the fields that comprise that partition or index. The tables refresh schedule and refresh strategy should be described.
    c. The description should use correct spelling and grammar. If necessary, grammar and spell-check functionality of outside programs should be utilized to ensure this.
    d. It is likely that comments inherited from the underlying database will be insufficient by themselves to accurately provide the robust description required. Preference should be given to a complete, verbose, understanding of the table rather than to brevity.
7. The usage property will be set for all query items in the DSQS using the following conventions:
    a. Set everything to "Attribute" except:
        i. Identifiers should be used for:
            1. Fields that are joined on.
            2. Fields that are part of a key.
        ii. Facts should be used for:
            1. Fields that are typically numeric, additive or semi-additive, and that the users have requested they want aggregated (e.g. dollar amounts). Many numeric data types in the database are not "facts".
8. The aggregate property will be set for all query items using the following conventions:
    a. If the field's usage is set to "attribute" or "identifier", then set to "Unsupported".
    b. If the field's usage is set to "fact", then set to the aggregate specified by the user.
    c. All fields whose usage is "fact," must have an aggregate function specified for them.
9. The format property will be set for all query items using the following conventions:
    a. Formats should be set for all fields
        i. Datetime fields should have their properties set as follows:
            1. Format Type dropdown = Date
            2. Date Style = Short

3. Display Years = Show Century
4. Display Months = Two Digits
5. Display Days = Two Digits

ii. VARCHAR fields should be formatted to “Text.”

iii. Currency fields should have their properties set as follows:
1. No. of Decimal Places = 2
2. Use Thousands Separator = Yes

iv. Number fields should have their properties set as follows:
1. No. of Decimal Places = 0 for integer values, as dictated by business needs for floating point numbers.
2. Use Thousands Separator = Yes
3. Otherwise zero precision (points after decimal) for integers
4. Zero Value Character = Zero (unless business needs dictate otherwise.)

v. Percentage:
1. Percentage Symbol = “%”
2. Use Thousands Separator = Yes
3. No. of Decimal Places = as dictated by business needs.
4. Zero Value Character = Zero (unless business needs dictate otherwise.)

vi. Comply with any user requests for a specific currency type or precision when dealing with numeric fields.

10. The “Description” property of all Query Items should be set and contain a description of that Query Item. The description should convey a rich understanding of the Query Item’s data and should be written using the following guidelines:

    a. The description should be understandable to a layman and assume no prior knowledge of the data contained in the Query Item or its associated business processes. This description may be read by end-users and future model maintainers who are not subject matter experts, and have no experience with, the Query Item or its data.

    b. If the Query Item is part of a hierarchy (for dimension tables) that hierarchy should be described. If the Query Item is part of an index or partition that should be indicated. If the Query Item contains a coded value, then the valid options of that code should be provided assuming there are a reasonable number of possible values. If the Query Item allows NULL values this should be indicated.

    c. The description should use correct spelling and grammar. If necessary, grammar and spell-check functionality of outside programs should be utilized to ensure this.

    d. It is likely that comments inherited from the underlying database will be insufficient by themselves to accurately provide the robust description required. Preference should be given to a complete, verbose, understanding of the Query Item rather than to brevity.

11. If embedded SQL is used in a Data Source Query Subject, a textual description of that SQL should be entered into the “Model Comments” property for that Data Source Query Subject. Appropriate comments should also be left in the SQL.

<u>LAYER 2: Database Abstraction Layer</u>

1. The purpose of this layer is to isolate higher layers from changes to the underlying database structure.
2. The Database Abstraction Layer will consist of Model Query Subjects (MQS), folders, embedded filters and data level security.

3. Within the Database Abstraction layer a Database Schema Folder should be created for each database schema used in the model. (These folders should match those in the Database Layer of the model.)
4. The Database Schema Folder will be named the same as the schema it represents.
5. All MQS should reside in the same Database Schema Folder as the schema their underlying Database Source Query Subjects are sourced from.
6. One Model Query Subject per Data Source Query Subject in the Database Layer (layer one). Multiple Model Query Subjects derived from the same Data Source Query Subject should only be created if required by differing embedded filters or data-level security.
7. Model Query Subjects in this layer should never be sourced from more than one Data Source Query Subject in the Database Layer (layer one).
8. Embedded filters will be added to Model Query Subjects at this layer as needed.
9. Data-Level ("row level") security (from the menu: Actions/Specify Data Security…) will be added to Model Query Subjects at this layer as needed.
10. If an Embedded Filter or data-level security is applied to a Model Query Subject, a textual description of that filter/security should be entered into the "Model Comments" property for that Model Query Subject.
11. Determinants should be set for all Model Query Subjects which represent dimensions (in dimensional modeling terms) and whose data reflects a hierarchy. (E.g. city, county, state, country, etc.)
12. Properties for Query Subjects and Query Items that were set in Layer 1 of the model should be inherited and/or re-entered as necessary.

<u>LAYER 3: Alias Layer</u>

1. The purpose of the Alias Layer is to A) maximize reuse of the Model Query Subjects (MQS) in the Database Abstraction Layer (layer 2), B) allow the same MQS to be joined in multiple ways, and C) allow multiple developers to work on the same model through use of Framework Manager branch and merge functionality.
2. The Alias Layer will consist of folders and shortcut objects to Model Query Subjects in the Database Abstraction Layer (layer 2). Alias shortcuts should be used except when reference shortcuts are explicitly required to maintain join paths.
3. The Alias Layer will contain one or more Relationship Namespaces. Use of Relationship Namespaces allows the same Model Query Subjects to have different relationships established for them depending upon user requirements. Some examples of when to use multiple Relationship Namespaces include A) multiple star-schema groupings sharing the same underlying tables, B) different reports requiring different join paths for the same tables, and C) alternate join paths to speed SQL performance and D) subdivision of a large subject area into smaller logical units.
4. Relationship Namespaces will be subdivided into Relationship Folders. Each star-schema/snowflake grouping of shortcut objects should be contained within its own Relationship Folder.
5. Every Model Query Subject in the Presentation Layer (layer four) will be sourced from its own Relationship Folder. There should be a one-to-one correlation of Model Query Subjects in the Presentation Layer (layer four) with Relationship Folders in the Alias Layer. All the shortcuts which comprise a single Model Query Subject in the Presentation Layer (layer four) should come from the **same, corresponding,** Relationship Folder.
6. All shortcuts in this layer must reside within a Relationship Namespace and relationship folder.

7. As the point of Relationship Namespaces is to isolate different sets of relationships, joins between Relationship Namespaces should occur **only** if dictated by business requirements. When joins between Relationship Namespaces do occur, they should occur through use of reference shortcuts rather than having actual relationship objects created for them. Example: Two star-schemas each in its own Relationship Namespace might be joined through a common dimension table. By using a reference shortcut to that dimension table in each Relationship Namespace, a relationship between those two star-schemas is established.
8. All the shortcut objects in a single Relationship Namespace should be joined to all other shortcuts in that same Relationship Namespace. This may be an indirect path through several other tables but there should never be a situation where a shortcut has no relationships defined for it or a Cartesian product is possible.
9. Relationship Namespaces will be named: "*Relationship Name (Subject Area Name)*", where *relationship name* is descriptive of the relationships being defined. Example: Paid Encounters (Expedited Medical Encounters), Denied Encounters (Expedited Medical Encounters).
10. Relationship Folders will be named the same as the Model Query Subject they represent in the Presentation Layer (layer four) of the model.
11. Prior to establishing a relationship, the records in the tables to be joined should be evaluated for any data issues that would cause records to be dropped if an inner join is used. If necessary, SQL queries and verification of the ETL should be used to validate joins. Potential data issues which should be evaluated include, but are not limited to, the following:
    - NULL values in a field used in the join.
    - Invalid values in a field used in the join.
    - Values in a field that do not resolve to their corresponding dimension.
    - Default values ('#," "?", "-1", "-2") in a field that do not exist in their corresponding dimension.
    - Problems joining fields of differing data types. (E.g. a VARCHAR field joined to a NUMBER field where the VARCHAR cannot be cast because some records contain non-numeric characters.)
12. Preference should be given to establishing relationships as inner joins with outer joins being used only if required by business rules. Outer joins should **not** be used to make up for data issues in the underlying tables. Attempts to resolve these underlying data issues should be made prior to utilizing an outer join.
13. Depending upon the nature of a relationship, the same underlying database table can serve as a fact table in one join and a dimension table in another. Cognos uses the cardinality defined for a relationship when optimizing SQL and performing aggregation. Therefore, the cardinality of relationships in the Cognos Model shall be as follows, regardless of the actual cardinality of data between the tables.

| Type of Relationship | Cardinality | Notes |
|---|---|---|
| Fact to Dimension | Fact 1..N ↔ 1..1 Dimension<br>Fact 1..N ↔ 0..1 Dimension | • Usually occurs when joining shortcut objects within the same Relationship Folder. |
| Fact to Fact | Fact 1..N ↔ 1..N Fact<br>Fact 0..N ↔ 0..N Fact | • Usually occurs when joining the main fact tables in two different Relationship Folders or Relationship Namespaces. |

14. Relationships should be manually defined by the model developer; they should not be imported from the Data Warehouse as the relationships may change depending upon the business requirements.
15. Joins will be named: "*Shortcut Name <##> Shortcut Name*" where *Shortcut Name* is the name of the shortcut being joined to one another. "**<**" and "**>**" are required literals and ## designates inner or

outer joins using the literal "-" for inner join and "0" for outer join. Example: T_RCPT_MISC_FACT <0-> T_ELIGTY_FACT represents that every T_RCPT_MISC_FACT record has a matching T_ELIGTY_FACT record, but not every T_ELIGTY_FACT record has a matching T_RCPT_MISC_FACT record.

LAYER 4: Presentation Layer

1. The purpose of the presentation layer is to allow the same underlying data-model (layers one and two) and relationships (layer three) to be presented in various ways depending upon business requirements.
2. Objects in the presentation layer will be visible to the end user. Therefore, the names for **all** objects in the Presentation Layer (including the name of the Presentation Layer itself) should be approved or provided by the end user prior to model development.
3. All the Model Query Subjects in a single Presentation Namespace should be capable of being used with all the other Model Query Subjects in that same Presentation Namespace without risk of a Cartesian product. In practice, this means all the Model Query Subjects in a Presentation Namespace are sourced from the **same** Relationship Namespace in layer 3. (It may be possible to source a Presentation Namespace from multiple Relationship Namespaces in layer 3 if care is taken in developing the Relationship Namespaces in such a way that there is a join path between them. However, this practice is discouraged unless required by business needs.)
4. All Model Query Subjects created in this layer must be created within a Presentation Namespace.
5. There should be a one-to-one correlation of Presentation Namespaces to Relationship Namespaces found in the Alias Layer (layer three) of the model. Model Query Subjects within a Presentation Namespace should be sourced from shortcuts objects (Relationship Folders) in their corresponding Relationship Namespace in the Alias Layer (layer three) of the model.
6. The Query Items composing Model Query Subjects in this layer shall be sourced **only** from the shortcuts established in the Alias Layer (layer three). Query Items should never be sourced directly from Query Subjects in the first two layers of the model.
7. A single Model Query Subject in this layer can contain Query Items sourced from multiple shortcuts (and ultimately database tables) in layer 3. (This is unlike the Query Items of a Model Query Subjects in layer one and two which all ultimately come from the same database table.) However, all of these shortcuts should come from their corresponding Relationship Folder in the Alias Layer (layer three). There should be a one-to-one correlation of Model Query Subjects in the Presentation Layer with Relationship Folders in the Alias Layer and all the shortcuts which comprise a Model Query subject should come from the **same** Relationship Folder.
8. Model Query Subjects will be named: "*Business Name*", where *Business Name* is the business name as agreed upon or supplied by the end user. Model Query Subjects that contain dimensional data should end in "Details". Do not use the words "Fact" or "Dimension" in naming any folder.
9. The Query Items within Model Query Subjects of this layer will be given business names as provided by the user and should adhere to the following rules:
    a. For query items previously defined in other Model Query Subjects or Projects, use the identical name unless it conflicts with the other naming standards (below) or the user requests otherwise.
    b. Names used must match the metadata provided by the end user.
    c. Phrase indicators as questions. Example: "Food Stamps Certified?" instead of "Food Stamp Certification Status Indicator."

d. Use “Begin” and “End” within the names of dates. Date fields should end in the word ‘date.’ Example: “Service Begin Date” instead of “Date Service Start.”

e. For Query Items that contain a textual description for a coded value, end the name with the word “Description.” Example: Category of Assistance Description

f. For Query Items that contain a coded value, end the name with the word “Code.” Example: Category of Assistance Code

g. Avoid use of the word ‘of.’ Example: “Capitation Days” instead of “Number of Capitation Days.”

h. Avoid acronyms and abbreviations. Example: “Medical Assistance Transportation Program” instead of “MATP.”

i. Objects with Object-Level security defined on them will adhere to the following rules:

   i. Query items will be named using all capital letters. Example: “SOCIAL SECURITY NUMBER” instead of “Social Security Number.”

   ii. Container objects (Folders, Query Subjects, and Namespaces) will be named in all capital letters **only if all objects** (Query Items, Folders, etc.) within that container use Object-Level security.

10. Query Items in a Model Query Subject should be placed above any Folders in that Query Subject.

11. Stand-Alone Filters should use the following rules:
    a. Stand-Alone Filters should only filter upon Query Items in the 4th layer or Shortcuts in the 3rd layer, of the model. Stand-Alone Filters should never directly reference Query Items in the first two layers of the model.
    b. Stand-Alone Filters should reside in the same Namespace of the Presentation layer as the Query Items to which they are associated.
    c. For complex Stand-Alone Filters a textual description of that filter should be entered into the “Model Comments” property of that filter object.
    d. All Stand-Alone Filters should be arranged alphabetically (top to bottom when viewed in the “Project Viewer” pane of Framework Manager) in the Namespace they reside within. Stand-Alone Filters should be placed in a folder titled “Filters” above any Model Query Subjects or Calculations.

12. Stand-Alone Calculations should use the following rules:

    a. Stand-Alone Calculations should only be performed on Query Items in the 4$^{th}$ layer of the model; they should not reference Query Items in other layers.

    b. Preference should be given to creating calculations as Query Items within the Model Query Subjects they are associated with rather than creating Stand-Alone Calculations.
    c. Stand-Alone Calculations should reside in the same Namespace of the Presentation layer as the Query Items they are calculating.
    d. For complex Stand-Alone Calculations a textual description of that calculation should be entered into the “Model Comments” property of that calculation object.
    e. All Stand-Alone Calculations should be arranged alphabetically (top to bottom when viewed in the “Project Viewer” pane of Framework Manager) in the Namespace they reside within. Stand-Alone Calculations should be placed in a folder titled “Calculations” below Stand-Alone Filters and above Model Query Subjects.

13. Object-Level (“column level”) security (from the menu: Actions/Specify Object Security…) will be added to Query Items, Model Query Subjects, and Namespaces in this layer as needed.

    a. Objects with Object-Level security defined on them should adhere to the naming conventions defined above.

b. Objects with Object-Level security will have the phrase "(SECURE)" added to their tooltip. This phrase should come before any other tooltip text entered by the package developer.

14. Properties for Query Subjects and Query Items that were set in Layer 2 of the model should be inherited and/or re-entered as necessary.

### Packages

Cognos BI Framework Manager provides the capability to create multiple packages from one model. This is preferable to creating multiple models for the same physical database. The naming standard for packages is below. This name will be visible to developers and users utilizing the package via the Cognos BI report development tools such as Report Studio.

1. Packages will be named: "*Business Name Package Type* PACKAGE" where *Business Name* has been approved by the end user. *Package Types* are: Ad Hoc, Metrics, OLAP, and Reporting. A space should separate each word. Example: Expedited Medical Encounters Reporting Package.
2. Packages should only make visible to the user objects in the 4th, Presentation, layer of the model. The first three layers of the model should never be visible in the package.

## Framework Manager Modeling Standards for OLAP Cubes:

The following standards apply to all models which will be used as a data-source (package or a report) for OLAP cube building. These standards are in addition to those described above.

1. When new subject areas are added to the Data-Warehouse, any requirements for cubes associated with those subject areas should be considered. If this consideration is given in the requirements phase of a project there will be minimal need to create new tables/star-schema groupings in either the Data-Warehouse or Framework Manager models after the fact.
2. All cubes should be based around star-schema/snowflake groupings with a central fact table surrounded by one of more dimension tables. These schemas should be optimized to support cube builds. (This is in contrast to a more normalized relational model as would be found in a transactional system.)
3. The measures of a cube should be sourced from fields in the central fact table of this star-schema/snowflake grouping. Measures should not be derived from the dimension tables of the star-schema grouping.

The following guidelines should be used when determining which Framework model to include these star-schema (snowflake) groupings in:

| Guidelines | Standard |
|---|---|
| • The tables used to build a cube represent a new subject area **not yet described** by an existing Framework Manager model. | • Create a new model following Framework Manager modeling standards. |
| • The tables used to build a cube are part of a subject area **already described** by an existing Framework Manager model.<br>**-AND-**<br>• The cube's star schema groupings are **not already described** in their own relationship namespace within the Framework Manager model. | • As required add the table(s) necessary to support the cube(s) to the existing Framework Manager model following the Framework Manager modeling standards. |
| • The tables used to build a cube are part of a subject **area already described** by an existing Framework Manager Model.<br>**-AND-** | • Use existing Framework Manager package |

| | |
|---|---|
| • The cube's star schema groupings are **already described** in their own relationship namespace within the model. | |

LAYER 3: Alias Layer

1. A single Relationship Namespace should be created in the Alias Layer (third layer) of the model. This Relationship Namespace will contain the relationships (Relationship Folders and alias shortcut objects) which comprise all the cubes associated with this model.
2. This Relationship Namespace will be named: "OLAP Cubes (*Subject Area Name.)"* Example: OLAP Cubes (Expedited Medical Encounters), OLAP Cubes (LIHEAP).
3. Each star-schema/snowflake grouping used to define a cube should be described in its own Relationship Folder. These Relationship Folders should all reside in the "Cubes…" Relationship Namespace of this model.
4. These Relationship Folders should be named the same as the business name of the cubes they represent.
5. The relationships defined in this Relationship Folder should be optimized to support cube builds rather than general "Ad-hoc" reporting.

LAYER 4: Presentation Layer

1. A single Presentation Namespace should be created in the Presentation Layer (fourth layer) of the model. This Presentation Namespace will contain all the cubes associated with this model.
2. This Presentation Namespace will be named "*Subject Area Name* OLAP Cubes." Example: Expedited Medical Encounters OLAP Cubes, LIHEAP OLAP Cubes.
3. Within this presentation namespace a single Model Query Subject should be created for each star-schema/snowflake grouping used to describe a cube. (Every Model Query subject should correspond one-to-one with a Relationship Folder in the Alias Layer (third layer) of the model.) All the fields (facts and dimensions) used to comprise a cube should be included in this Model Query Subject.
4. These Model Query Subjects should be named the same as the business names of the cubes they represent. Example: Individuals Cube, LIHEAP Payments Cube. (If multiple cubes are derived from the same Model Query Subject (star-schema/snowflake grouping) then an appropriately descriptive name should be chosen.)
5. Query Items within Model Query Subjects of this layer will be given business names which match those used in the OLAP cube.

Packages:

1. The "…Cubes" Presentation Namespace should not be visible in general Ad-Hoc or reporting packages unless a valid business reason exists otherwise. Rather they should be published in their own OLAP package(s).
2. Cube packages should be named "*Business Name* OLAP Package." Example: "Encounters OLAP Package."

## Naming Conventions:

- All Framework Manager objects must follow the naming conventions defined below.
- *Italicized* text indicates a variable name. Normal text indicates a literal.

| Project Level Objects | | | | |
|---|---|---|---|---|
| **Object** | **Naming Convention** | **Examples** | **Notes** | **Layer** |
| Project (CPF) Filename | *Subject Area Name* Project | Expedited Medical Encounters Project.CPF | | NA |
| Data Source | *Subject_Area_Name_Data_Source_Schema_Name* | Expedited_Medical_Encounters _EDWP_EDW | Use underscores instead of spaces. | NA |
| Model Namespace | *Subject Area Name* Model | Expedited Medical Encounters Model | | NA |
| Package | *Business Name Package Type* Package | Expedited Medical Encounters Reporting Package | See the rules for naming packages above. | NA |

| Layers (Layer Namespaces) | | | | |
|---|---|---|---|---|
| **Object** | **Naming Convention** | **Examples** | **Notes** | **Layer** |
| Database Layer Namespace | Database Layer (*Subject Area Name*) | Database Layer (Expedited Medical Encounters) | | NA |
| Database Abstraction Layer | Database Abstraction Layer (*Subject Area Name*) | Database Abstraction Layer (Expedited Medical Encounters) | | NA |
| Alias Layer | Alias Layer (*Subject Area Name*) | Alias Layer (Expedited Medical Encounters) | | NA |
| Presentation Layer | *Subject Area Name* | Expedited Medical Encounters | The Subject Area Name should be approved by the end user. | NA |

| Shortcuts & Relationships | | | | |
|---|---|---|---|---|
| **Object** | **Naming Convention** | **Examples** | **Notes** | **Layer** |
| Shortcuts | *MQS_Name+Alias_Clarifier* | T_COUNTY_DIM<br>T_COUNTY_DIM+Recipient<br>T_COUNTY_DIM+Medical_Provider | A clarifier is only required if an alias to the same MQS will be used multiple times within the same Relationship Folder.<br>Use underscores instead of spaces. | 3 |
| Relationship (join) | *Shortcut Name* <##> *Shortcut Name* | T_RCPT_DIM <0-> T_COUNTY_DIM | See the rules for naming Relationships in Layer 3 of the model. | 3 |

## Query Subjects

| Object | Naming Convention | Examples | Notes | Layer |
|---|---|---|---|---|
| Data Source Query Subjects (DSQS) | *Database_Table_Name* | T_COUNTY_DIM | Use underscores instead of spaces. | 1 |
| Data Source Query Subject (DSQS) derived from hard-coded SQL | *Database_Table_Name*+SQLn | T_COUNTY_DIM+SQL1<br>T_COUNTY_DIM+SQL2 | Use numbers to distinguish DSQS if the same DSQS is filtered multiple ways. | 1 |
| Model Query Subject in Layer 2 | *DSQS_Name* | T_COUNTY_DIM | Use underscores instead of spaces. | 2 |
| Model Query Subject in Layer 2 with filter applied | *DSQS_Name*+F*n* | T_COUNTY_DIM+F1<br>T_COUNTY_DIM+F2 | Use numbers to distinguish MQS if the same MQS is filtered multiple ways. | 2 |
| Model Query Subject in Layer 2 with data-level security applied | *DSQS_Name*+S*n* | T_RCPT_DIM+S1<br>T_RCPT_DIM+S2 | Use numbers to distinguish MQS if the same MQS has data-level security applied in it multiple ways. | 2 |
| Model Query Subject in Layer 4 | *Business Name* | Recipient County Details | The business name should be approved by the end-user. | 4 |
| Model Query Subject in Layer 4 containing objects with Object-Level Security | *BUSINESS NAME* | RECIPIENT DETAILS | Model Query Subjects should be capitalized **only** if **all** enclosed objects (Folders and Query Items) have Object-Level security applied to them. | 4 |

## Query Items

| Object | Naming Convention | Examples | Notes | Layer |
|---|---|---|---|---|
| Query Items in a DSQS | Corresponding field name in the database table they are sourced from. | CDE_COUNTY<br>NAM_COUNTY | | 1 |
| Query Items in a Layer 2 MQS | Query Item names should match the Query Item names in the DSQS objects they are sourced from | CDE_COUNTY<br>NAM_COUNTY | | 2 |
| Query Items in a Layer 4 MQS | *Business Name* | Recipient ID<br>Program Begin Date<br>Program End Date | See the rules for naming Query Items in the description of the Presentation Layer. | 4 |
| Query Items in a Layer 4 MQS with Object-Level Security | *BUSINESS NAME* | SOCIAL SECURITY NUMBER | These names should be in all capital letters. | 4 |

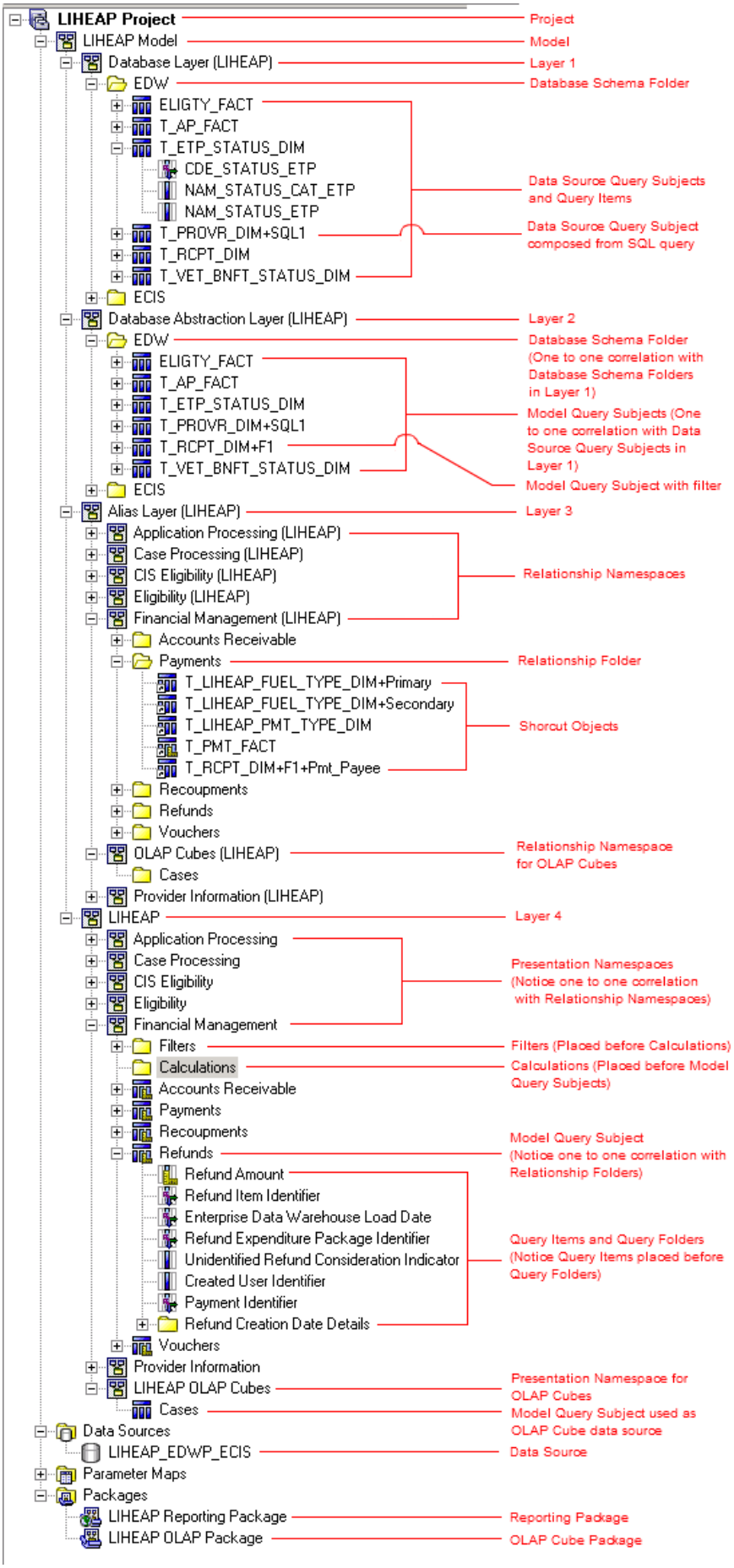
## Folders

| Object | Naming Convention | Examples | Notes | Layer |
|---|---|---|---|---|
| Database Schema Folder | *Database Schema Name* | EDW<br>ECIS | The Database Schema Folder name should be the same as the underlying database schema it represents. | 1 and 2 |

| | | | | |
|---|---|---|---|---|
| Relationship Folder | *Business Name* | Payments<br><br>Recipients | The Relationship Folder name should be the same as the business name of the Model Query Subject it represents in the Presentation Layer (fourth layer) of the model. | 2 (within Relationship Namespaces) |
| Folder in Layer 4 containing objects with Object-Level Security | *BUSINESS NAME* | RECIPIENT DETAILS | Folders should be capitalized **only** if **all** enclosed objects (Folders, Model Query Subjects, Query Items, etc.) have Object-Level security applied to them. | 4 |

| Namespaces | | | | |
|---|---|---|---|---|
| **Object** | **Naming Convention** | **Examples** | **Notes** | **Layer** |
| Relationship Namespace | *Namespace (Subject Area Name)* | Paid Encounters (Expedited Medical Encounters)<br>Denied Encounters (Expedited Medical Encounters). | The Namespace name should be descriptive of the relationships being defined in this Namespace. | 3 |
| OLAP Cube Relationship Namespace | OLAP Cubes *(Subject Area Name)* | OLAP Cubes (Expedited Medical Encounters)<br>OLAP Cubes (LIHEAP) | | 3 |
| Presentation Namespace | *Presentation Name* | Paid Expedited Medical Encounters<br>Denied Expedited Medical Encounters | The Presentation Namespace Name should be approved by the end user. | 4 |
| OLAP Cube Presentation Namespace | *Subject Area Name* OLAP Cubes | Expedited Medical Encounters OLAP Cubes | | 4 |
| Namespace | *Business Name* | County Details<br>Recipient Details | The Business Name should be approved by the end user. | 4 (within Presentation Namespaces) |

| Misc. Objects | | | | |
|---|---|---|---|---|
| **Object** | **Naming Convention** | **Examples** | **Notes** | **Layer** |
| Stand-Alone Filters | *Descriptive Business Name* Filter | Active Records Only Filter | | 4 |
| Stand-Alone Calculations | *Business Name* | Sum Of Claim Amounts | | 4 |




An visual example of the layout and naming conventions for the sample LIHEAP project.

## Exemptions from this Standard:

All existing production models/packages as of June 26, 2009 are exempt. For all new additions or major modification to existing exempt models/packages the exemption does not apply.

All exemptions to this standard must be approved on a case-by-case basis by EKMS after appropriate justification for an exemption has been provided

## Refresh Schedule:

All standards and referenced documentation identified in this standard will be subject to review and possible revision annually or upon request by the DHS Information Technology Standards Team.

## Standard Revision Log:

| Change Date | Version | Change Description | Author and Organization |
|---|---|---|---|
| 07/09/2008 | 1.0 | Initial Creation | Larry Leitzel, EKMS |
| 6/24/2009 | 2.0 | Additions to give new directions on creating layers | EKMS Team, EKMS |
| 03/02/2010 | 3.0 | Additions to apply “lessons learned” and provide for OLAP Cube development. | EKMS Team, EKMS |
| 09/28/2010 | 3.1 | • Updated requirements for date time fields to be formatted as MM/dd/yyyy.<br>• Updated wording on determinants. | EKMS Team, EKMS |
| 11/04/2010 | 3.2 | Added naming conventions for object level security. | EKMS Team, EKMS |
| 11/27/2013 | 3.3 | • Added requirement for dimensional modeling and star-schemas.<br>• Added requirements for Description property of Query Subjects and Query Items to provide additional metadata to end users.<br>• Added package governor settings.<br>• Updated Format property of Query Items to handle additional data types. | Bryan Porter, EKMS |
| 02/17/2016 | 3.4 | • Added wording indicating standards are being updated.<br>• Added wording indicating a Dimensional Model is no longer optional. | Bryan Porter, EKMS |