Base Data

Base data is the term used throughout MONAHRQ to refer to data that is required for MONAHRQ to function. This includes collections such as area population data, cost to charge ratio data, and DRG values.

Data Sets

External Data Sets Included in MONAHRQ's Core

Name	Description	Versioning	Source
AreaPopulationStrats	Area Population by County	Year	US Census
CGCAHPSMeasureLookup	CG-CAHPS measure types (e.g.: yes/no questions)	Version Num.	AHRQ
ConsumerPriceIndex	Used for inflation adjustments in cost trending reports	Year	St. Louis FRED
CostToChargeRatio	Cost to charge ratio calculated using HCRIS cost reports	Year	CMS
CostToChargeToDRG	Relates cost to charge ratios to DRGs	Year	CMS
Counties	Master list of all counties	Version Num.	US Census
DRG	Master list of MS-DRG codes	Version Num.	CMS
DXCCS	Master list of Clinical Classifications (CCS) based on diagnosis codes	Version Num.	AHRQ
DXCCSCategories	CCS Groupings	Year	AHRQ
DXCCSLabels	CCS labels	Year	AHRQ
EDNationalTotals	Benchmark data for Emergency Department utilization reports	Year	AHRQ HCUPnet
HospitalCategories	Facility type	Year/Month	CMS
HRR	Hospital Referral Regions master list	Year	Dartmouth Atlas
HRRtoPopulationStrats	Population by HRR regions; updated annually	Year	Dartmouth Atlas
HSA	Hospital Service Areas master list	Year	Dartmouth Atlas
HSAtoPopulationStrats	Population by HSA regions; updated annually	Year	Dartmouth Atlas
ICD9toDXCCSCrosswalk	Maps ICD9 codes to DXCCS codes	Year	AHRQ
ICD9toPRCCSCrosswalk	Maps ICD9 codes to PRCCS codes	Year	AHRQ
ICD10toDXCCSCrosswalk	Maps ICD10 codes to DXCCS codes	Year	AHRQ
ICD10toPRCCSCrosswalk	Maps ICD10 codes to PRCCS codes	Year	AHRQ
IPNationalTotalsDRG	Benchmark data for Inpatient utilization reports (MS-DRG based reporting). Updated annually.	Year	AHRQ HCUPnet
IPNationalTotalsDXCCS	Benchmark data for Inpatient utilization reports (DX-CCS based reporting). Updated annually.	Year	AHRQ HCUPnet
IPNationalTotalsMDC	Benchmark data for Inpatient utilization reports (MDC based reporting). Updated annually.	Year	AHRQ HCUPnet

IPNationalTotalsPRCCS	Data for Inpatient utilization reports (PR-CCS based reporting). Updated annually.	Year	AHRQ HCUPnet
MDC	Major Diagnostic Categories master list	Version Num.	CMS
MSDRG	MS-DRG master list	Version Num.	CMS
NHCAHPSMeasureLookup	Describes CAHPS question types that correspond to NH-CAHPS measures	Version Num.	AHRQ
NHProviderToLatLong	Coordinates for nursing home providers	Version Num.	Medicare Data
POSHospitals	Hospital master list. Updated annually.	Year/Month	CMS
PRCCS	Clinical Classifications (CCS) master list, based on procedure code	Version Num.	AHRQ
PRCCSCategories	Categories used in the PRCCS master list	Year	AHRQ
PRCCSLabels	Descriptions for PRCCS categories	Year/Month	AHRQ
ProviderSpecialities	Descriptions for provider specialties	Version Num.	CMS
States	Master list of all states	Version Num.	US Census
ZipCodeToHRRandHSA	Zip code to HRR and HSA crosswalk. Update as needed.	Year	Dartmouth Atlas
ZipCodeToLatLong	Coordinates for zip codes	Year	US Census
ZipCodeToPopulationStrats	Population by ZIP code. Update as needed.	Year	US Census

Data Sets Internal to MONAHRQ

These include enumerations and internal lookup data. These data sets are defined in the `Monahrq.Infrastructure` assembly.

Misc. Data Sets

Source data for the following data sets may be found in the `Resources\BaseData` directory of the `Monahrq.Infrastructure` project and the MONAHRQ program directory.

Name	Description	Versioning
MeasureFilters	Unknown; possibly unused	Version Num.
MeasureTopics	Measure topics and topic categories	Version Num.
Menus	Information about menus and their structure used in generated websites	Version Num.
BaseWebsitePage	Listing of website pages included in generated websites	Version Num.
BaseWebsitePageZone	Additional metadata about included pages	Version Num.

Enumeration Data Set Types

The enumeration types below are defined in the `Monahrq.Infrastructure.Entities.Domain.Wings` namespace.

Name	Description	Values[1]
Admission Source	Patient origin	`ER, OtherHospital, OtherFacility, LegalSystem, Routine, Missing`

[1] Values exclude common members such as `Missing`, `Exclude`, and `Retain`

Name	Description	Values[1]
Admission Type	Discharge classification	`Emergency, Urgent, Elective, Newborn, Trauma, Other`
Definite	Indicator of agreement for CAHPS survey	`YesDefinitely, YesSomewhat, No`
Discharge Disposition	Disposition of patient on discharge	`Routine, ShortTerm, NursingFacility, ImmediateCare, OtherFacility, HomeHealthCare, AMA, Deceased, DischargedAliveDestUnknown`
Gender	Patient's gender	`Male, Female`
Hospital Trauma Level	Trauma level	`NotTraumaCenter, Level_1, Level_2, Level_3, Level_4, Level_1_2, Level_1_2_3`
How Often	Indicator of frequency for CAHPS survey	`Never, Sometimes, Usually, Always, Inapplicable, NoAnswerGiven`
Number of Times	Indicator of frequency for CAHPS survey	`Never, Once, ThreeTimes, Inapplicable, NoAnswerGiven`
Point of Origin	Patient origin	`NonHealthCare, Clinic, TransferFromOther, TransferInternal, TransferExternal, ER, LegalSystem, OtherHealthAgency, ReadminFromSame, TransferDistrict, TransferAmbulatory, TransferHospice`
Primary Payer	Primary payer for patient	`Medicare, Medicaid, Private, SelfPay, NoCharge, Other`
Race	Patient's race	`White, Black, Hispanic, AsianPacificIsland, NativeAmerican, Other`
Rate Provider	Provider's CAHPS ratings lookup	`0 (Worst) – 10 (Best)`
Rating	Nursing Home's CAHPS ratings lookup	`0 (Worst) – 10 (Best)`
Sex	Patient's sex	`Male, Female`
Yes/No	Yes or No CAHPS question responses	`Yes, No Inapplicable, NoAnswerGiven`

Versioning Strategies

Every base data set type identifies installed versions of itself by querying the `SchemaVersions` table for rows with matching `Name` and `FileName` columns. From the rows that remain, MONAHRQ uses one of three versioning strategies to uniquely identify the installed version(s) of each base data set:

- `DefaultBaseDataVersionStrategy`[2]
  Compares the `Version` column to the version of the data set known to MONAHRQ. In many cases, the version number is extracted from the filename that contains the data for the base data set.

- `MonthAndYearBaseDataVersionStrategy`
  Compares the `Month` and `Year` columns to the date of the data set known to MONAHRQ. The month and year are extracted from the filename that contains the data for the base data set.

[2] Refer to the `Monahrq.Infrastructure.BaseDataLoader` namespace for more information about built-in base data sets, versioning strategies, and their base types

- `YearOnlyBaseDataVersionStrategy`
  Compares the `Year` column to the date of the data set known to MONAHRQ. The year is extracted from the filename that contains the data for the base data set.

The implementation of all versioning strategies may be found in the `Monahrq.Infrastructure.BaseDataLoader` namespace.

Selecting a Versioning Strategy

The versioning strategy selected for a particular type of base data is largely a matter of preference and intended use. If only one version of a base data set is needed at any time, `DefaultBaseDataVersionStrategy` is probably the most appropriate choice. However, if the same base data type requires multiple data sets (e.g.: one per year), then one of the date-centric version strategies may be more appropriate.

Base Data Type Definition

Base data types are defined by CLR classes that implement `IBaseDataImporter`[2]. This interface provides several key members:

- `VersionStrategy`
  A method for detecting the currently installed version(s) of a base data set; see *Versioning Strategies*

- `DatabaseTableName`
  The name of the table in the database that contains the base data set

- `LoadData()`
  The method that imports data into the MONAHRQ database

The interface provides several other members which may be useful to implementers. Refer to the source code for more information.

Choosing a Base Class

Several implementations of `IBaseDataImporter` are provided to cover common use cases:

- `BaseDataEnumImporter<TEntity, TKey, TEnum>`
  Imports an enum values from type `TEnum` as the entity type `TEntity`. Existing data is truncated.

- `BaseDataDataReaderImporter<TEntity, TKey>`
  Creates records of `TEntity` for every row in an `IDataReader`; the `IDataReader` is created by opening a CSV file prefixed with the `Fileprefix` property. The `ImportType` property determines how existing data is handled.

- `BaseDataSqlBulkImporter<TEntity, TKey>`
  Performs a SQL bulk import of a CSV file prefixed with `Fileprefix` using a format file `FormatFile` to map CSV columns to SQL columns. The `ImportType` property determines how existing data is handled.

- `BaseDataImporter<TEntity, TKey>`
  Generic type for importing `TEntity` instances having a primary key of type `TKey` from a file; provides some helper methods but does not provide any method implementations. Nor does it implement any handling of existing data.

Lifecycle

The lifecycle of an `IBaseDataImporter` implementation is managed by `BaseDataModule` in `Monahrq.Wing.ReportingEntities`. When MONAHRQ starts, `BaseDataModule` performs the following steps for each detected `IBaseDataImporter`:

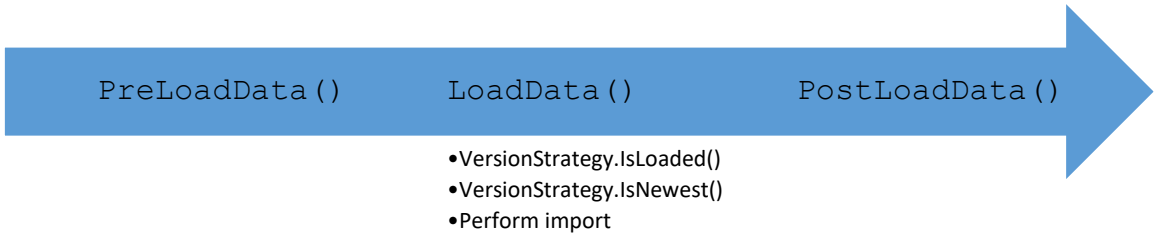

Figure 1 Base Data Lifecycle

1. Call `PreLoadData()`
2. Call `LoadData()`
   a. Check `VersionStrategy.IsLoaded()` and `VersionStrategy.IsNewest()`; exit if true
   b. Perform data import
3. Call `PostLoadData()`

Exporting

All implementations of `IBaseDataImporter` should be exported using `DataImportContracts.BaseDataLoader`, as follows.

```
[Export(DataImportContracts.BaseDataLoader, typeof(IBasedataImporter))]
public class TopicsStrategy : BaseDataCustomImporter<Topic, int>
{
```

Figure 2 Sample ExportAttribute for IBaseDataImporter implementation

For additional information about how `IBaseDataImporter` implementations are consumed, refer to `BaseDataModule` in `Monahrq.Wing.ReportingEntities`.

Sample Custom LoadData() Implementation

The following is a simplified version of the `LoadData()` implementation from `HospitalRegistryStrategy`.

```
var session = DataProvider.SessionFactory.OpenSession();
// look for an existing HospitalRegistry record
if (session.Query<HospitalRegistry>().Any())
    return;
// create a new record and save it
session.SaveOrUpdate(new HospitalRegistry(1)
{
    Name = GetType()
        .Assembly
        .GetCustomAttribute<AssemblyDescriptionAttribute>()
        .Description
});
// save a SchemaVersion record, too
session.SaveOrUpdate(base.VersionStrategy.GetVersion(session));
session.Flush();
```

Figure 3 Simplified LoadData() implementation from HospitalRegistryStrategy

Alternative Uses for Base Data Importers

The `DefaultBaseDataVersionStrategy` could be used by a .NET Wing to perform database schema updates after an upgrade. This may be useful if the Wing defines a Target and that Target's type definition changes from one version to the next.