WebSphere® Business Monitor

IBM

Version 6.0

Database services

Note!
Before using this information and the product it supports, be sure to read the general information under "Notices and Trademarks" on page 77

Sixth Edition (December 2005)

This edition applies to Version 6 of IBM WebSphere Business Monitor product (5724-M24) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You can send to the following address:

Cairo Technology Development Center (CTDC)
Business Integration Product Development
IBM WTC – Egypt Branch
Pyramids Heights Office Park, Building C10
Cairo – Alexandria Desert Road, km. 22
P.O. Box 166 El-Ahram, Giza, Egypt

Include the page number or topic related to your comment.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Managing databases

Managing WebSphere® Business Monitor databases is particularly important during the installation and deployment phases of WebSphere Business Monitor.

Managing databases includes the following tasks:

- Creating databases
- Setting the appropriate databases configuration
- Creating static and dynamic database tables and indexes
- Deploying the generated replication scripts
- Deploying the cube view metadata definitions
- Maintaining databases

Note: Throughout the WebSphere Business Monitor documentation, the underlying schemas of various databases are discussed. Some of the Schema Generator output provides insight into the nature of the WebSphere Business Monitor database schemas. While the DBA will likely make use of this information for maintaining and tuning the datastore, this information should not be taken as the definition of a public API. It is entirely possible and likely that future versions of WebSphere Business Monitor will change this underlying scheme. Customers should not develop infrastructure that assumes that backward compatibility to these schemas will be maintained in future releases. Custom code written to make use of the WebSphere Business Monitor databases may, and probably will, not be compatible with future product releases.

The following information will help you plan and prepare for managing the WebSphere Business Monitor databases.

Databases configuration and replication

Data management plays a major role in WebSphere Business Monitor.

The database architecture of WebSphere Business Monitor supports the following requirements:

- Isolating runtime processing on the datastore from the client access datastore to maintain the appropriate processing rate
- Being able to perform updates on the client access datastore and still respond quickly to client queries
- Optimizing the access to historical datastore for analytical and multidimensional reporting purposes

The usage patterns of the data in the WebSphere Business Monitor databases differ according to the using component. The data is used by two major components: the event processor and the client dashboards. This difference in usage makes it essential to split the database of the event processing from the database of the dashboards. The data can be further classified into information associated with a business measures model and information about processing events.

The dashboards display two types of data, recent data and historical instances data. The number of recent instances is very small compared to the number of historical instances. The queries performed on the recent instances need to be extremely fast and must not be affected by the large number of historical instances. The two types of data have been split into two databases, Runtime and Historical. To enhance performance, the architecture supports all functions with the following:

- A database serving as the definitions container for the business measures models. It also stores information about other databases.
- A database serving as a transactional database and used by the event processor.
- A database that acts as a near real-time analysis database, supporting analysis queries without affecting the transactional server. It is used by the dashboards.
- A database that supports multidimensional analysis over the history of the transactions. It is used by the dashboards to view historical data.

The WebSphere Business Monitor databases are divided into four different databases:

- **Repository**: Stores business measures models and event definitions. It also stores the schemas, names and host names of State, Runtime, and Historical databases.
- **State**: Stores the current state of the running process instances and the values of the business measures associated with each process instance. It is used for event processing by WebSphere Business Monitor server.
- **Runtime**: The State and Runtime database store pretty much exactly the same information. The Runtime database differs only by how some data is stored, how recent the data is, and how long the data stays. The data in the runtime database stays at least 24 hours longer than it does in the State database. The purpose of the Runtime database is to allow user to perform near real time analysis without affecting the event processing done by the WebSphere Business Monitor server. The Runtime database serves client queries on recent instances. It stores the runtime information of the business measures group for efficient reporting. It is used for dashboards viewing.

- **Historical database**: Stores the information of the completed instances and the current state of the running instances in a star schema, for historical, multidimensional reporting. It is used for dashboards viewing.

Two databases store the monitored events and the Adaptive Action Manager data. These databases are used internally by the WebSphere Business Monitor. No information related to process instances or metrics is stored in them.

- **Emitter**: Store the events emitted from the engines. The emitters database tables reside in the engines databases.
- **Action catalog**: Stores the events that are defined as situations and actions that the Adaptive Action Manager must perform on them. It is created during installation.

WebSphere Business Monitor databases

WebSphere Business Monitor uses four databases to store event data and business measures model metadata. The four are: Repository, State, Runtime, and Historical databases.

Repository database

The Repository database contains the metadata describing the currently deployed business measures models as well as information about the other WebSphere Business Monitor databases. The Repository database contains the history of the deployed models. There is only one Repository database per WebSphere Business Monitor installation.

The Repository database is used by the Launchpad, which populates it with the database attributes for the State, Runtime, and Historical databases. These attributes are the database name, database schema and host names of the database server. They are used by the other WebSphere Business Monitor components to access the State, Runtime, and Historical databases at runtime. The Repository database is also populated during the import of the business measures model.

The Repository database is used by the following components:

- **Administrative console**

  A business measures model is imported through WebSphere Business Monitor administrative console. The definitions of the processes and events of this imported model are stored in the Repository database. When import is complete, the business measures model is considered to be deployed. After the model is imported, the definitions of the processes and events are available to other WebSphere Business Monitor components for retrieval.

  The Repository database is also used by the Schema Generator. The Schema Generator needs to know the schema name that should be used in its database artifact generation. And also when a user modifies a previously deployed business measures model and attempts to regenerate the schema for it, the Schema Generator checks the existence of artifacts in the Repository database before generating the change management artifacts.

- **Dashboards**

  The dashboards have a set of views that show data from different perspectives. Some of these views are populated with data from the Runtime database, and some with data from the Historical database. To allow the user to configure and parameterize these views, the views need to retrieve WebSphere Business Monitor metadata from the Repository database. Some of the views need to compose queries against the DB2® Alphablox Cubes. The composition of these

queries requires metadata about the dimensions, measures, and cube names that will be retrieved from the Repository database. Also the dashboards displays the process diagrams for business processes which are stored in the Repository database.

- **WebSphere Business Monitor server**

  The WebSphere Business Monitor server uses the Repository database to retrieve the definitions of processes and events.

State database

The State database stores information about running instances. This information includes metrics, business measures, and key performance indicators (KPIs) values. It is optimized for heavy transaction workloads. There is only one State database per WebSphere Business Monitor installation.

Each process instance requires two tables in the State database to store metrics, business measures, and KPIs. The structure of these tables is as dynamic as the structure of the process instance. Each business measure is represented by a separate column in one of the two tables. Depending on the options selected during the building of the business measures models, much or all of the information in the State database is replicated to the Runtime database.

The State database is used by WebSphere Business Monitor server. At runtime, the WebSphere Business Monitor server inserts, retrieves, and updates the information of processes instances that reside in the State database, according to the processed events.

The State database stores the following information:

- Information about business measures group, which is a part of the data in the imported business measures models.
- The running process instances that are created while the WebSphere Business Monitor is running.
- The event entries of the running processes. The event entry is the event data that is received for updating a specific business measures group.

Runtime database

The Runtime database is similar in structure to the State database. It receives replicated information from the State database about the current status of all running processes as well as the final status of recently completed or failed processes. This information is used by WebSphere Business Monitor dashboards. The Runtime database is also used by the Adaptive Action Manager to store alert notifications. There is only one Runtime database per WebSphere Business Monitor installation.

The Runtime database stores:

- Alert notifications sent by the Adaptive Action Manager to the dashboards
- Process data
- Metrics values

The information in the Runtime database is replicated from the State database.

The Runtime database is used by theWebSphere Business Monitor dashboards. The dashboards retrieve the running or recently completed instances data required to

populate the views from the Runtime database. The dashboard views use the Runtime database for analytical purposes, so it is optimized for query processing and aggregate query processing.

Historical database

The Historical database stores all completed and running process instances. It is used by the dashboards for enhanced data analysis using DB2 Alphablox. There is only one Historical database per WebSphere Business Monitor installation. The data in the Historical database is never deleted.

The Historical database should only contain two years worth of historical data. This is one of WebSphere Business Monitor product requirements. As mentioned before, the historical data is never deleted automatically, so the DBA is responsible for deleting the data that is greater than two years old. The Historical database stores the information regarding long-running instances as well as completed instances. This information is stored as star schemas rather than in the flat transactional forms used in the State and Runtime databases. The Historical database is optimized for aggregated and long running querying. It is used byDB2 Alphablox in dashboards views to provide advanced multidimensional reports.

The information in the Historical database is replicated from the Runtime database.

In the Historical database, each process instance has its own set of tables. Unlike the State and Runtime databases, each set of tables is a star schema that supports multidimensional reporting.

The Historical database contains dynamic tables that are created according to the deployed business measures model. The Schema Generator generates the Historical database schema, which is used to create dynamic tables, and Cube Views™ definitions.

The Historical database is used by the WebSphere Business Monitor dashboards. The dashboards retrieve the data required to populate some views from the Historical database. For example, the Reports view focuses on analyzing data extracted from the Historical database.

The Historical database contains the following information:

- The data of various versions of the process instances that are running or terminated.
- The data of the completed process instances that are stored in the Runtime database. All completed instances remain in the Runtime database for 24 hours. 24 hours is the default retention policy which can be modified as part of the data movement service configuration. After this data is replicated to the Historical database, it is deleted from the Runtime database to speed up performance.

Databases creation and deployment

The WebSphere Business Monitor databases are created by the Launchpad during installation. Before you create the databases, you first need to plan for their creation and deployment.

If WebSphere Business Monitor databases are dropped or damaged irreparably after installation, the database administrator (DBA) can manually re-create the databases by executing the creation scripts that are saved in

<monitor_installation_dir>\install\mondb. The DBA can also uninstall the databases using the Launchpad, and to create the databases, the DBA should first drop the database manually from the DB2 and then recreate the databases again using the Launchpad.

Preparation of database artifacts deployment

Before you start creating WebSphere Business Monitor databases using the Launchpad, you need to plan for the databases. Planning includes allocating databases sizes, preparing backup strategies, configuring data movement services, and setting table spaces and buffer pools parameters as well as determining the settings for the database instances and the individual databases.

During installation, the Launchpad creates the State, Runtime and Historical databases and the database objects that will be used for administrative purposes. In addition to these objects, the Schema Generator created a set of business measures model dedicated database objects (like tables). The Launchpad creates a set of default table spaces and buffer pools for the State, Runtime and Historical databases. These default table spaces are referenced in a table space configuration file and are designed to allow users to quickly get up and running for testing and proof of concept scenarios. To avoid performance problems and resource constraints, it is essential to plan ahead how tables will be spread across table spaces, which containers and buffer pools will be used by the table spaces.

During installation, the databases are created and only the static table definitions are created. During the artifacts generation, the dynamic tables are assigned to table spaces in the State, Runtime and Historical databases based on a customizable text configuration file. WebSphere Business Monitor is shipped with a default configuration file which is located in the <Monitor_install_dir>\install\mondb directory. This default configuration file maps all tables to exactly one table space of the appropriate size. To support ad-hoc deployments, Launchpad created during installation a set of table spaces (with 4KB, 8KB, 16KB and 32KB page size) that correspond to the entries in the default configuration file. The following example shows an excerpt from the default table space configuration file that ships with WebSphere Business Monitor:

```
#
# State database
#
db2.state.Default.TABLE.4K.0=DSDFLTTS4
db2.state.Default.TABLE.8K.0=DSDFLTTS8
db2.state.Default.TABLE.16K.0=DSDFLTTS16
db2.state.Default.TABLE.32K.0=DSDFLTTS32
```

This simple configuration is used during artifacts generation, all tables that need to be created in the State database that fit into a table space with 4KB page size would be assigned to `DSDFLTTS4`. Tables with a page size of 8 KB would be stored in a table space named `DSDFLTTS8`, and tables with a page size of 16 KB to `DSDFLTTS16`. As table space requirements may vary (depending on the complexity of the model for which the data is to be stored as well as the amount of data), it is advisable not to use the default table space configuration settings in a test or production environment. Planning ahead and determining a suitable storage strategy ensures good performance.

More advanced configuration file settings can be used to map tables to table spaces not only based on the page size but also on what kind of data is going to be stored. Check the example configuration file for more information.

To determine which table space a table will be assigned to, the Schema Generator performs the following tasks:

- Determine table type.
- Calculate the minimum page size necessary to store at least one row of data.
- Identify available table spaces for this type; if found, locate the next available table space and use it. If no table space was found, continue.
- Identify available Schema Generator default table spaces; if found, locates the next available table space and use it. If you did not find any available table space for this type, then continue.
- Assign table to database default table space (by not specifying a table space clause during table creation).

  Note: Deployment will fail if no default table space has been defined in the databases for the required page size.

You can edit the table space configuration file using any text editor or create a new one. Use the general configuration tab in the Schema Generator Administration console to force Schema Generator to use an alternate configuration file.

Note: The Schema Generator does not actually create any table spaces for the entries in this configuration file. This needs to be done manually prior to deploying the generated database artifacts. Artifacts deployment will fail if a table was assigned to a table space that does not exist.

Database artifacts deployment

You deploy database tables after you have created WebSphere Business Monitor databases with the Launchpad. In the deployment phase, the Schema Generator is configured to generate the artifacts that will be deployed to complete the databases setup. The databases are then ready to be populated with data.

Note: The databases are created once. And for each business measures model, more database tables are added to a database.

Databases schema generation

WebSphere Business Monitor databases schemas are based on business measures models. These schemas are generated by the Schema Generator.

The business measures model is created using Business Measures editor, It is an annotated model, which contains the metadata of the business model. Using the Business Measures editor, the user can define what should be monitored: contexts, key performance indicators (KPIs), metrics, and business situations. The business measures model produces monitoring information through events. After the business measures model is completed, it is exported to WebSphere Business Monitor as an annotated XML file in zip format for use by Schema Generator.

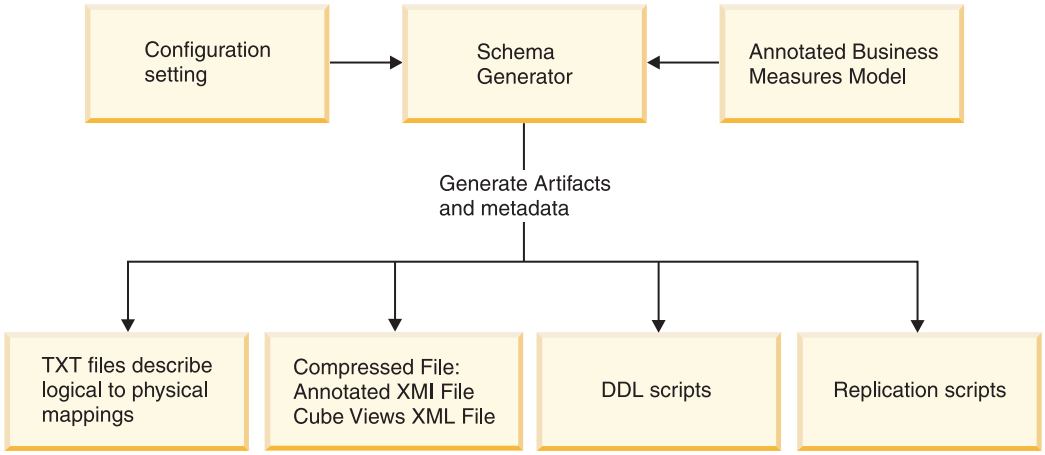
The Schema Generator is part of the WebSphere Business Monitor administrative console. Typically this tool will be configured and used by a database administrator (DBA). It takes the business measures model as an input; then generate the database artifacts.

The following are the generated artifacts:

- The database definition language (DDL) scripts suitable for creating the database tables. For each generated DDL file (state.ddl, runtime.ddl, datamart.ddl), a corresponding text file is created (stateMapping.txt, runtimeMapping.txt,

datamartMapping.txt). These files contain a description of the metrics and processes that the physical database artifacts (tables, columns) represent.

- DB2 Cube Views definitions, which describe the Historical database data in a star schema format. The cube views definitions is suitable for import into the DB2 OLAP center.
- Replication scripts that enable replication between the State, Runtime, and Historical databases. For each of these databases, the Schema Generatorcreates a compressed file containing all the deployment artifacts necessary to set up replication across the three databases. Typically these artifacts will be distributed and deployed by a DBA as per the instructions in "Deploying data movement services" on page 44.



The database tables in WebSphere Business Monitor database components consist of two types:

- There are static database tables that are created once at installation time. These tables are shared across all business measures models and do not depend on any single business measures model.
- There are dynamic database tables that are dependent on a business measures model that is imported into the WebSphere Business Monitor administrative console. The dynamic database tables schemas are unique for each business measures model. Any changes to the business measures model associated with the dynamic tables will result in a change management scenario. For more information about change management scenarios, refer to "Change management and artifacts generation" on page 34.

Cube views definitions

The Schema Generator creates a DB2 Cube Views XML file. The database administrator (DBA) imports this XML file into the DB2 OLAP Center.

The Schema Generator generates a Cube Views XML file based on a business measures model. The business measures model contains information that helps to describe measures and dimensions. You can also describe what aggregations can be applied to the measures.

For each process in the business measures model, a cube and a cube model are created. As well, a cube and a cube model are generated for the activities associated with a process. Each cube model and cube contain some predefined measures and dimensions that are generated automatically.

Each cube model and cube contain three built-in measures:

- **Elapsed duration**: Has a defined aggregation function **'avg'**.
- **Working duration**: Has a defined aggregation function **'avg'**.
- **Instances count**: Has a defined aggregation function **'count'**.

The following dimensions are automatically generated:

- **CreationTime**: The time when a process instance was created
- **StartTime**: The time when a process instance was started
- **State**: Contains all of the possible states (string values) a process instance may be in, such as started, running, or completed
- **TerminationTime**: The time when the process instance was terminated

These time-based dimensions use a commonly defined dimension that is provided (DIM_TIME). It has three predefined levels: year, month, and day.

At modeling time, you can define your own business measures. The business measures that are created can be measures or dimensions. The WebSphere Business Modeler documentation describes in detail how to use the WebSphere Business Modeler to create measures or dimensions.

For further information on DB2 Cube Views, refer to the DB2 documentation.

Artifacts generation and deployment

The Schema Generator generates database and cube views artifacts based on each imported business measures model.

The database administrator (DBA) executes the artifacts as a step in the business measures model deployment phase. The following tasks must have been completed prior to deploying any business measures model:

- Participate in pre-planning exercises. This involves deciding about the topology, determining event rates, number of users supported, database instance and database parameters, buffer pools and table spaces, and a backup and recovery strategy. It is also important to a have a strategy for determining how and where generated artifacts are stored. It would be wise to be able to find a set of previously generated artifacts by model and version number and whether they were deployed. This is useful for both change management and support scenarios.
- Create the Repository, State, Runtime, and Historical databases using the WebSphere Business Monitor Launchpad. Creating a database includes creating a set of static database tables, setting up the State and Runtime databases to be replication sources, and also creating some other database objects like stored procedures and udfs which are specific to various components of WebSphere Business Monitor.
- Run the Schema Generator to generate business measures model related artifacts.

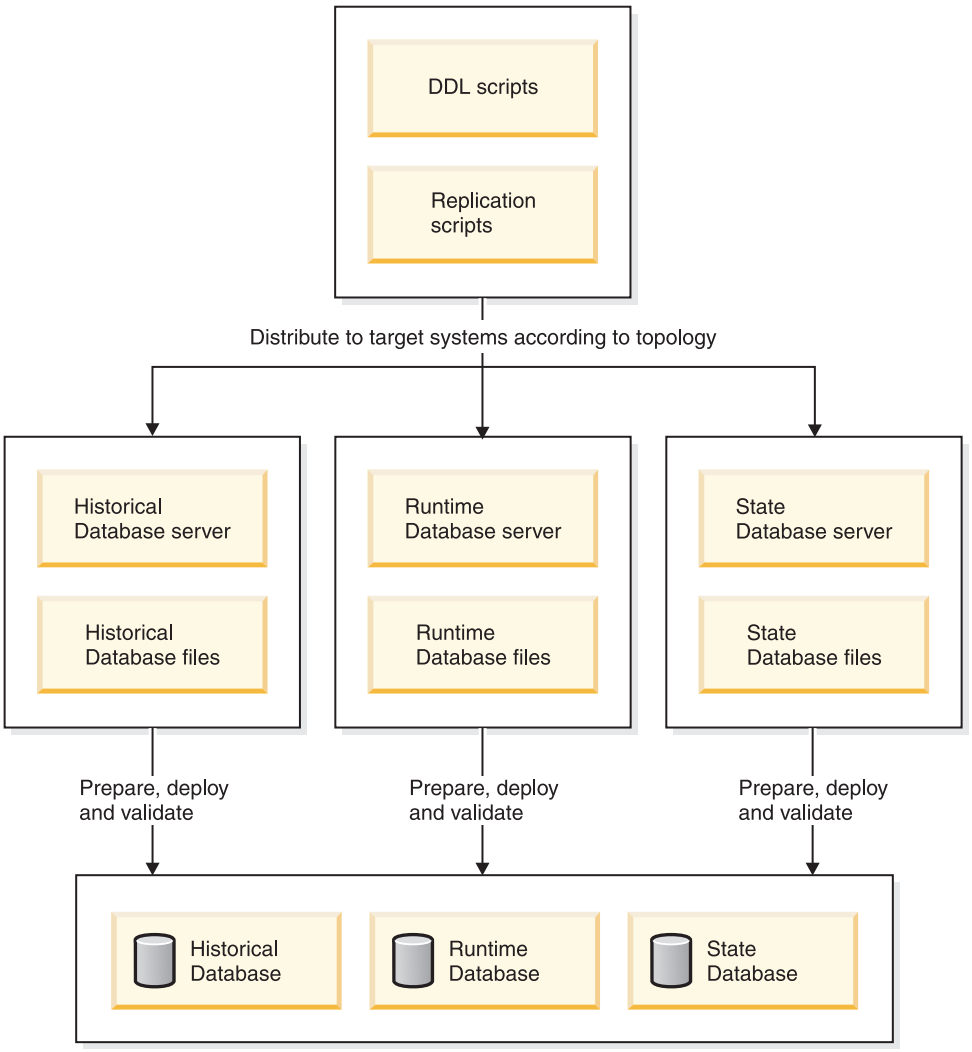
Note: It is recommended to back up all databases before you deploy any generated artifacts

After you have generated the artifacts, perform the following to deploy them:

- Deploy the generated DDL scripts to create the business measures model dynamic tables in the State, Runtime, and Historical databases.
- Execute the replication scripts to enable databases replication.

- Import the cube view definition file into DB2 Cube Views.

The following figure describes the artifacts deployment phase:



Artifacts customization

Under certain circumstances, it might be beneficial to alter the generated database artifacts to improve performance. In general, there are two approaches that can be taken: iterative improvement and ad-hoc improvement

Iterative table space mapping improvement

The number of tables that are created by the Schema Generator depends (among other facts) on the complexity of the business measures model. Thus it is initially difficult to determine how to optimally assign those tables to the table spaces. The following simple approach can help you to incrementally improve your table to table space mapping, as defined by the table space configuration file. For more information about database planning, refer to “Preparation of database artifacts deployment” on page 7. To improve table to table space mapping, you have to perform the following tasks:

1. Run the Schema Generator using the default or any other customized table space configuration file.

2. Extract the generated artifacts into a temporary directory and identify the number and type of tables that will be created in the database. For each business measures model, tables of the following types will be created:
   - **Context and activity**: Exist only in the State and Runtime databases and are accessed by the server and dashboards.
   - **CD**: Exists only in the State and Runtime databases. CD Tables usually contain a few more columns than there associated context or activity tables. The size of a particular CD table depends heavily on the number of transactions against its corresponding context or activity table, the ratio of update to insert transactions, and how often the associated apply component of a data movement service reads the transactions from the CD table and inserts them into a corresponding CCD table, and how often the CD table entries are pruned by the associated source life cycle component.
   - **CCD**: Exists only in the Runtime and Historical databases. A CCD table has the exact same structure as its corresponding CD table, and largely depends upon the same factors to determine its size. The one difference is that instead of being read by a apply component the transactions are read by an ETL component and entries are pruned by a target life cycle component.
   - **RM Internal Tables**: Exists only in the Runtime and Historical databases. These tables use pages with a maximum size of 4 KB.
   - **Fact and Dimension**: Exist only in the Historical database.
3. Modify the table space configuration file so that it contains:
   - A mapping for each table type
   - Multiple table type to table space mappings if a large number of tables would otherwise be assigned to the same table space

   Note:
   - Do not specify table-space declarations for table types that will not be created in a database because those table spaces will not be used.
   - Metrics map to table columns. The more metrics you define, the larger a table will be and the larger the page size that will be needed for its table space.
4. Create those table spaces (and buffer pools) before deploying the generated artifacts: The Schema Generator does not validate whether the table spaces declared in the configuration file exist because no database connection has been established. However, deployment will fail if the table spaces do not exist.
5. Re-Run the Schema Generation using the optimized table space configuration file.

   Note: Change the configuration file name in the Schema Generator administrative console configuration if you created a new table space configuration file.

Ad-hoc improvements

You can modify the generated artifacts by changing the following:

- Any index (add, alter, remove, assign to separate table space (DMS only)) to improve the database performance.

  Note: The creation of UNIQUE indexes needs to be carefully considered as this could lead to unexpected failures.

- Any table-space assignment (assign to a different table space, add table space assignment for indexes or large objects if using database-managed space (DMS) table spaces)
- Any comments on tables (not recommended, because the comments identify what each table and column represent)

In general, changes allowed do not alter the fundamental table schema or structure.

When you change the generated scripts, consider the following limitations:

- You cannot change any table name.
- You cannot change any column name.
- You cannot change any column data type.
- You cannot remove any column or any table.
- You cannot add any column to a table
- You cannot change any primary key for a table.
- You cannot change the nullability of a column.
- You cannot change the schema assignment for any table.
- You cannot add any new constraints such as uniqueness constraints or foreign key constraints.

Important: The changes made to the generated database artifacts are not taken into consideration the next time schema generation is performed for the same business measures model. For example, a user alters the generated artifacts for a business measures model ″Finance Model″ prior to deploying them. Subsequently the user alters the business measures model and regenerates all artifacts. In this case the user will have to modify the newly generated artifacts again because Schema Generator does not know about the modification made to the previously generated artifacts.

Note:

- The artifacts deployment requires Java™ version 1.4.2 or higher.
- The Java bin directory should be added to the system path before executing the replication scripts.

Database replication

Database replication technology is being used to move business measures model related data from the State database to the Runtime database and from the Runtime database to the Historical database.

To setup replication for a business measures model between the State, Runtime and Historical database, the following high level tasks need to be performed:

1. **Generate replication setup scripts**. The Schema Generator analyzes the business measures model for which replication has to be set up and generates a set of setup files.
2. **Distribute replication setup scripts**. These setup files have to be manually transferred to the machines hosting the State, Runtime and Historical databases.
3. **Execute replication setup scripts**. Each one of the setup files will upon execution create the necessary database objects and configure the utilities that move the data from one database to another.

Upon successful completion, the replication utilities can be started and are operational. The following sections provide an in-depth architectural view and explains some of the fundamental concepts used.

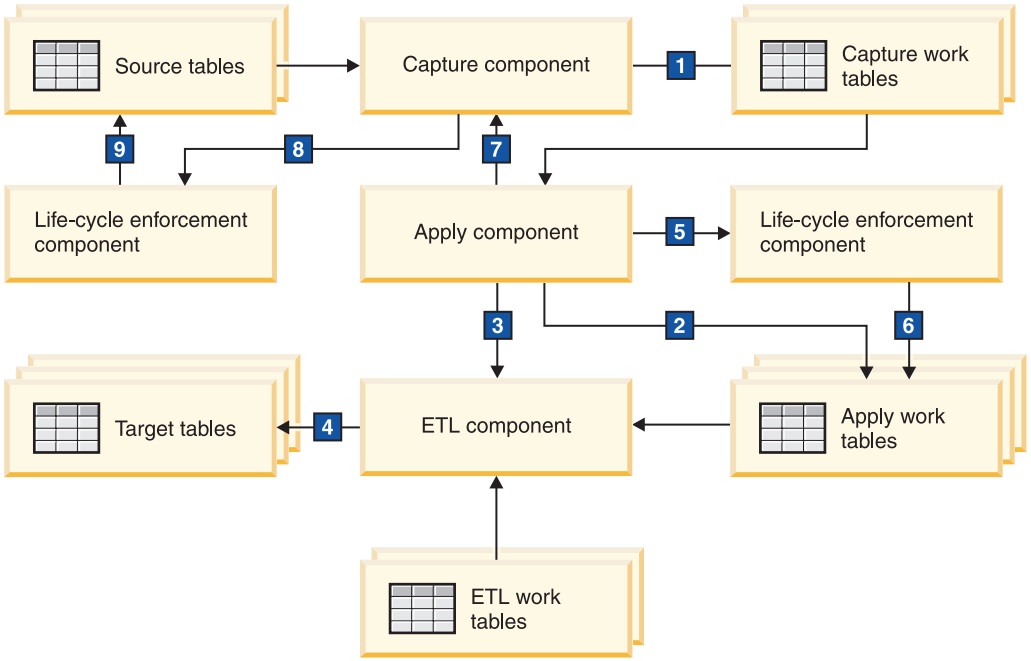
Data movement services

A data movement service enables an application to move data from a source database to a target database. Source and target databases can be homogeneous or heterogeneous: that is, residing on a single system or distributed across multiple systems. Aside from moving data, a service can transform data and provide basic data life-cycle functionality as required by the application.

Data movement services are implemented by five major components:

1. (Source) Capture component
2. (Target) Apply component
3. ETL (extract, transform, load) component
4. Source Life Cycle component
5. Target Life Cycle component

The Capture and Apply components work together to move the data from the source database to the target database. The ETL component performs any necessary data transformation if the data structures in the source database are different from the data structures in the target database.The following diagram illustrates the process flow within a data movement service:

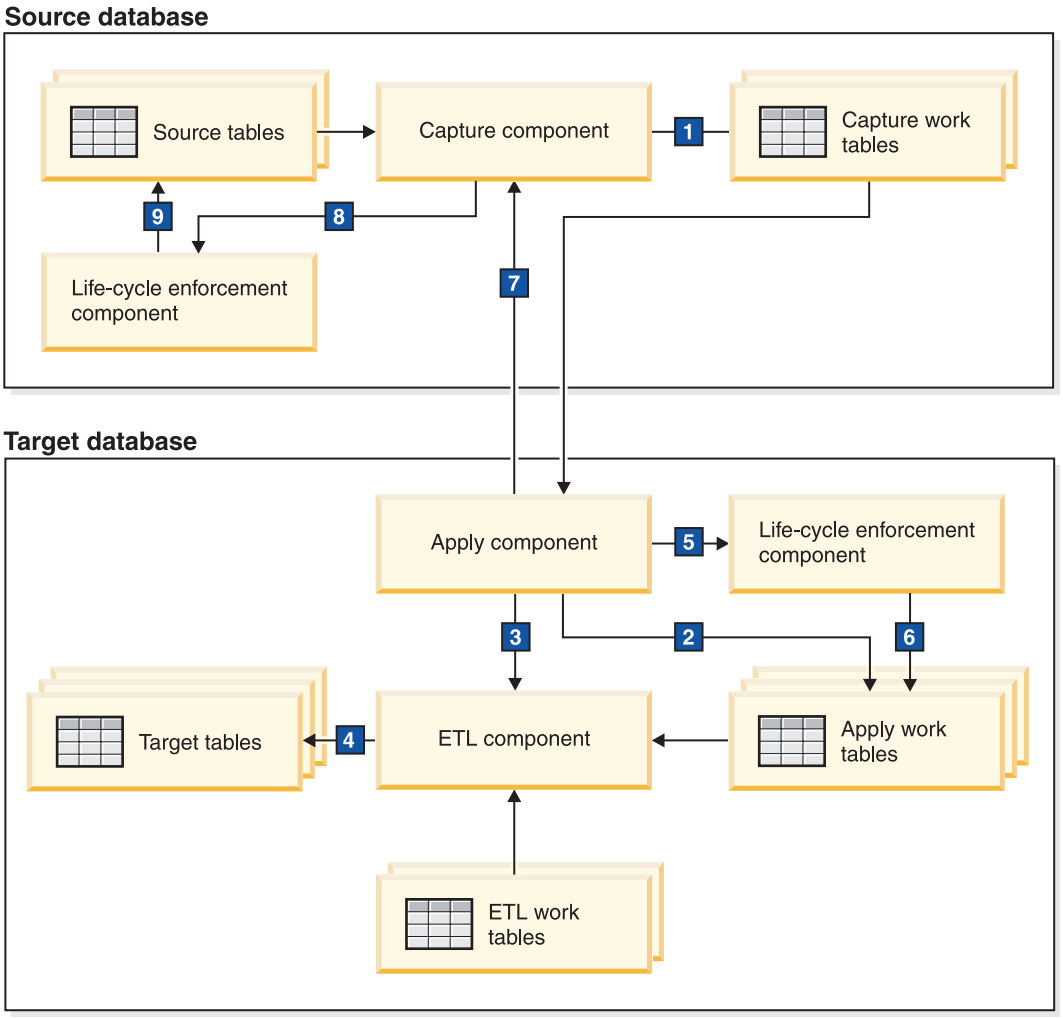


The data movement service flow follows these steps:

1. Data in source tables is stored and frequently updated, for example, by the Monitor server. The Capture component records, in the work tables, any data changes that are made to the source tables.
2. At predefined intervals, the changes are identified by the Apply component and recorded in the work tables.
3. After changes have been recorded successfully, the ETL component is invoked.

4. Using the data that is stored in the Apply work tables and predefined rules, the ETL component performs any necessary transformations. Data that has been successfully transformed is written to the target tables. Any incomplete or erroneous data is retained in another set of work tables for later processing.
5. Upon completion of the ETL processing, the Target Life Cycle component is activated.
6. Over time, large amounts of data can accumulate in the Apply work tables. Any data in those tables that has been successfully processed by the ETL component is removed by the Target Life Cycle component.
7. Once the data has been successfully copied to the target database, it is no longer needed and can be removed from the Capture work tables. The Capture component prunes the work tables periodically to reduce the resource contingencies.
8. Removal of data from the Capture work tables triggers the invocation of the Source Life Cycle component.
9. Any data that has been successfully processed, marked as ready for deletion, and has passed the Source Life Cycle retention Policy is removed from the source database.

The Capture component and the Source Life Cycle component usually reside on the source system; the Apply, ETL and Target Life Cycle components reside on the target system as shown in the following figure:



Within a data movement service, multiple instances of the components may be used, depending on the data structures used in the source and target database. The number of component instances is directly related to the number of business measures groups and the number of source and target tables within a business measures model. Each instance is uniquely identified. The following rules apply withinWebSphere Business Monitor:

- One Capture component instance is assigned to one business measures model project and captures changes for all source tables that belong to this business measures model project.
- One Apply component instance is assigned to one business measures model project and records changes that need to be applied to the target tables that belong to this business measures model project.
- One ETL component instance is assigned to one target table.
- One Source Life Cycle component instance is assigned to one source table.
- One Target Life Cycle component instance is assigned to one Apply work table.

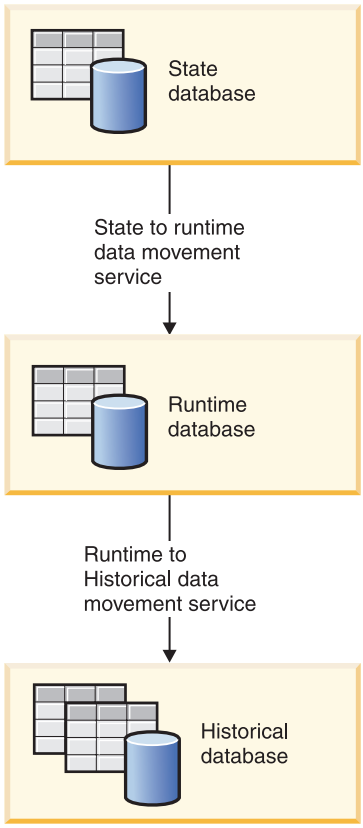
A component instance can be, for example, an executable program, a database stored procedure, or a database trigger.

Two instances of data movement services are used in WebSphere Business Monitor:

- State to runtime data movement service

- Runtime to historical data movement service.

The State to Runtime data movement service processes data that has been stored by the Monitor server in the State database and moves that data into the Runtime database where it can be accessed by the dashboard. The Runtime to Historical data movement service moves data from the Runtime database to the Historical database.The following diagram illustrates this movement:



The following information describes the default configurations for these services and how to configure, start and stop, and monitor them.

State to Runtime data movement service

The State to Runtime data movement service processes data that has been stored by the Monitor Server in the State database and moves that data into the Runtime database where it can be accessed by other WebSphere Business Monitor components and the Runtime to Historical data movement service.

The following default configuration applies to this data movement service:

- Changes in the Monitor Server source tables (State database) are continuously captured and recorded in work tables.
- Changes that have been recorded in those work tables are continuously propagated by the Apply component and applied to work tables in the Runtime database. Those work tables are not accessible by any other WebSphere Business Monitor component and are for internal use only.
- The Apply component synchronously invokes the ETL component every time new data needs to be processed. Depending on its schedule, which is initially set to every 5 minutes, the ETL component either processes data that is stored in the Apply work tables or remains inactive until it is scheduled to run. Increasing the

delay between scheduled runs results in a longer elapsed time between when data was stored in the State database by the Monitor Server and the time this data is published in the target tables in the Runtime database. Once it is in the Runtime database, the data can be accessed by other WebSphere Business Monitor components.

- Any data in the Apply work tables that has been successfully processed by the ETL component is removed by the Target Life Cycle component according to its schedule. By default, this component runs once every 24 hours. Increasing the scheduled delay causes the work tables to grow larger. Decreasing the delay can also cause contingency problems because multiple data service components might try to update and access the work tables concurrently.
- Data that has been successfully moved from the Capture work tables to the Apply work tables is automatically removed from the Capture work table by the Capture component every 5 minutes by default.
- Each time the Capture work tables are pruned, the Source Life Cycle component is invoked. This component is also schedule based. It only removes data from the source tables that has been marked as ready for deletion by the Monitor Server, if at least 5 minutes have passed since the last data pruning. If the Life Cycle component pruning interval is set to a value that is lower than the pruning interval of the Capture component, pruning is based on the Capture component pruning interval.

  For example: The Capture component pruning interval for work tables is set to 5 minutes, and the Source Life Cycle component schedule is set to 1 minute. Five minutes have to pass before the Capture program can start its pruning cycle. Because the Capture pruning routines are not activated for 5 minutes, the Life Cycle component is not invoked. After 5 minutes have passed, data is removed from the work tables, and the Source Life Cycle component is invoked and removes data from the source tables in the State database.

The default configuration can be changed.

Runtime to Historical data movement service

The Runtime to Historical data movement service moves data from the Runtime database to the Historical database where it remains until it is explicitly removed by the database administrator (DBA). Data that has been successfully moved into the Historical database is available for retrieval and analysis by other WebSphere Business Monitor components.

The following default configuration applies to this data movement service:

- Changes in the Runtime database tables are continuously captured and recorded in work tables. The Runtime database tables that are being monitored are the target tables that have been populated by the State to Runtime data movement service.
- Changes that have been recorded in those work tables are continuously propagated by the Apply component and applied to work tables in the Historical database. Those work tables are not accessible by any other WebSphere Business Monitor component and are for internal use only.
- The Apply component synchronously invokes the ETL component every time new data needs to be processed. Depending on its schedule, which is initially set to every 24 hours, the ETL component either processes data that is stored in the Apply work tables or remains inactive until it is scheduled to run. Increasing the delay between scheduled runs results in a longer elapsed time between when data was stored in the Runtime database by the State to Runtime data movement service and the time this data is published in the target tables in the

Historical database. Once it is in the Historical database, the data can be accessed by other WebSphere Business Monitor components.

Note: Because of the dependency on the invocation by the Apply component and configuration of the Apply component, an ETL component may not be able to process new data every 24 hours (or the current default delay). The schedule should rather be interpreted as ″do not process new data for at least 23 hours 59 minutes after the last processing cycle has finished.″

- Any data in the Apply work tables that has been successfully processed by the ETL component is removed by the Target Life Cycle component according to its schedule. By default, this component runs every 24 hours. Increasing the scheduled delay causes the work tables to grow larger. Decreasing the delay can also cause contingency problems because multiple data service components might try to update and access the work tables concurrently.
- Data that has been successfully moved from the Capture work tables to the Apply work tables is automatically removed from the Capture work table by the Capture component every 5 minutes.
- Each time the Capture work tables are pruned, the Source Life Cycle component is invoked. This component is also schedule based. It only removes data from the source tables in the Runtime database that has been marked ready for deletion by the Monitor Server and has remained at least 24 hours in the Runtime database. The default pruning interval is set to 5 minutes. If the Life Cycle component pruning interval is set to a value that is lower than the pruning interval of the Capture component, pruning is based on the Capture component pruning interval.

  For example: The Capture component pruning interval for work tables is set to 5 minutes, and the Source Life Cycle component schedule is set to 1 minute. Five minutes have to pass before the Capture program can start its pruning cycle. Because the Capture pruning routines are not activated for 5 minutes, the Life Cycle component is not invoked. After 5 minutes have passed, data is removed from the work tables, and the Source Life Cycle component is invoked and removes data from the source tables in the Runtime database.

The default configuration can be changed.

Data movement services administration

To start or stop a data movement service for a business measures model, you must identify the associated main component instances to enable or disable them.

All (Source) Capture component instances and all (Target) Apply component instances are considered main component instances. The ETL component instances and the Target Life Cycle component instances are dependent on the (Target) Apply component instances. The Source Life Cycle component instances are dependent on the (Source) Capture component and do not need to be started or stopped explicitly. In general, all (Source) Capture component instances in the source database and all (Target) Apply component instances have to be started or stopped by the user. Only after all of these instances have been started or stopped is a data movement service considered started (fully operational) or stopped.

A Capture component instance is equivalent to a DB2 Capture program, and an Apply Component instance is equivalent to a DB2 Apply program. Both programs can be started and stopped manually using scripts or through the use of scheduling tools or services, depending on the operating system your database

system has been installed on. During deployment, start and stop scripts that are ready for use are created. On Windows® systems, these scripts are batch files. On UNIX® systems, they are shell scripts.

Each one of the Capture programs needs to be started on the system that hosts the DB2 containing the source database (State database for the State to Runtime data movement service or Runtime database for the Runtime to Historical data movement service). The Capture component must have local access to the DB2 log files.

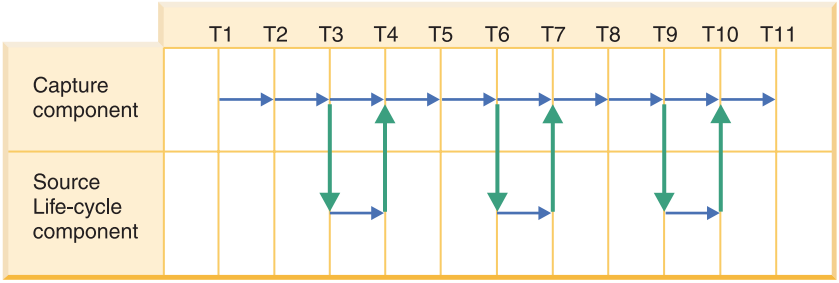
Note: All data changes in the source database will still be captured even if the Capture instances are not running. However, none of the captured changes is processed and applied to the target database tables until all instances are operational.

Data movement service configuration

The behavior and scheduling of each data movement services component can be configured to meet the different needs of a development, test, and production environment. Changing the configuration of one component can have a direct impact on the behavior of other components that are dependent on that component.

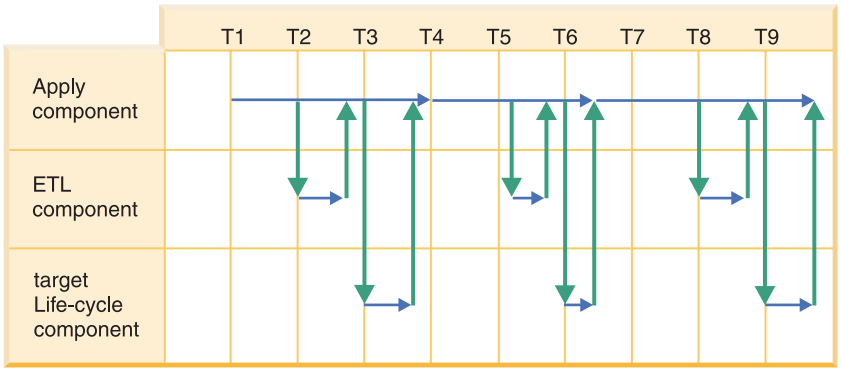
In general, there are two dependencies:

- The Capture component periodically invokes the Source Life Cycle component. If the Capture component is not running, no source life cycle enforcement is performed. The delay between life cycle component invocations is configurable. In the above figure, the Source Life Cycle component is invoked every 3 time

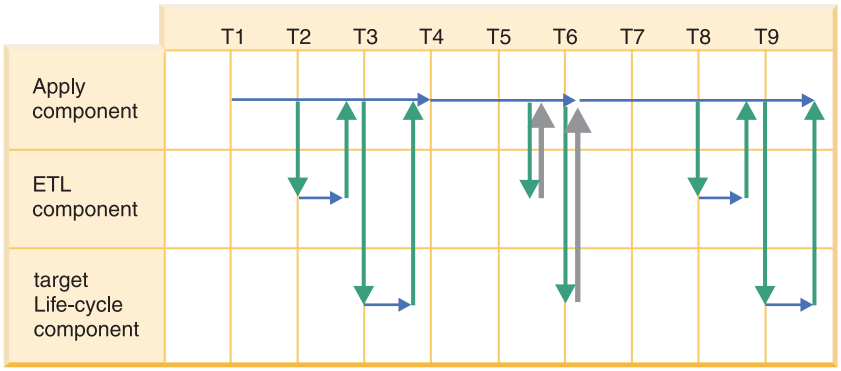


units, performs some activity, and then returns control to the Capture component, which continues processing.

- The ETL component and the Target Life Cycle component are invoked by the Apply component, after data has been successfully moved from the source database to the target database. The ETL and the Target Life Cycle components are only invoked when the Apply component is running.



Because the dependent components need to operate on different schedules than the component they are dependent on, an invocation does not necessarily result in execution. Instead, each dependent component checks its schedule upon invocation and returns control to the calling component if it is not yet time to perform any tasks. In the above example, ETL and Target Llife Cycle components might only be executed twice if the schedule for both components restricts them from being invoked more than once every 5 time units.



The ETL component (and Target Life Cycle component) is invoked and executed at T2 (T3 respectively). The next invocation occurs at approximately T6. Because fewer than 5 time units have passed since they last executed, control is immediately returned to the Apply component. Subsequent invocations at approximately T8 (T9 respectively) result in execution because more than 5 time units have passed. Each component is implemented by one or more component instances. You can configure each one of the instances separately to allow for more granular control.

Note: If changes are made, they take effect immediately, unless noted otherwise.

You can modify the default configuration for the Capture and Apply components by changing the appropriate control tables or by overriding them using command-line parameters in the start scripts. You can configure the ETL and life cycle enforcement components by updating one of the control tables.

You perform the following steps to customize data movement service components to meet the requirements of development, test, and production environments.

Configuring the (Source) Capture component instances

A Capture component instance is equivalent to a DB2 Capture replication utility. By default, this utility is configured to continuously capture changes to the source tables and record the changes in internal work tables. In general, you do not need to change the default configuration for the Capture component instances.

- **Identifying Capture component instances**.

  Multiple Capture component instances (that is, DB2 Capture utilities) are used to capture data associated with a business measures model. To determine which Capture utilities have been assigned to provide services for a business measures model, you must:

  – Identify the data movement service for which you want to change the Capture utility configuration.

  – Inspect the WBIRMADM.RMMETADATA metadata table in the State database (for the State to Runtime data movement service) or in the Runtime database (for the Runtime to Historical data movement service), and identify all Capture utility names (column SRC_RM_CAP_SVR_NAME).

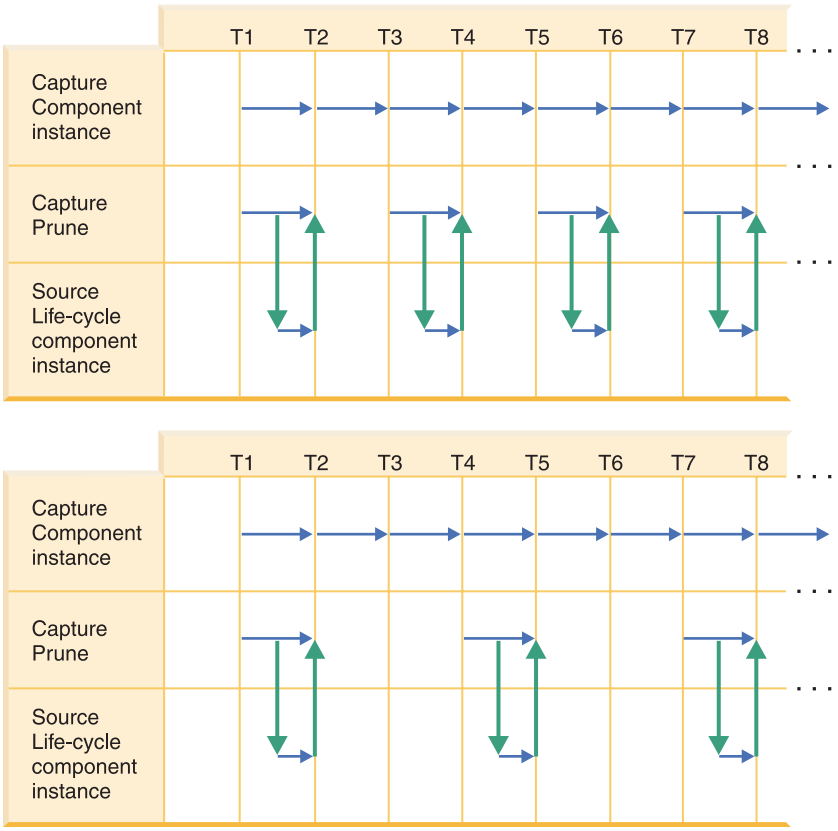
    Example: The query ″SELECT OM_NAME, SRC_TAB_NAME, SERVICE_NAME, SRC_RM_CAP_SVR_NAME FROM WBIRMADM.RMMETADATA WHERE SERVICE_NAME='State to Runtime' ″ may result in the following:

OM_NAME	SRC_TAB_NAME	SERVICE_NAME	SRC_RM_CAP_SVR_NAME
STEW_S	wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	State to Runtime	CAPTURE_1
STEW_S	wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE	State to Runtime	CAPTURE_1

    In the above example, the Capture utility CAPTURE_1 has been assigned to capture all changes made to the two source tables associated with business measures model STEW_S in the State database.

- **Changing the pruning interval of the Capture work table**.

  Capture utilities automatically prune their work tables every 300 seconds (the default for the prune_interval parameter) if auto pruning is enabled (autoprune parameter equals ″y″). Each pruning activity automatically results in an invocation of a Source Life Cycle component instance, which is implemented by a database trigger. Changing the pruning interval parameter for a Capture utility has a direct impact on how often the source tables are pruned by the Source Life Cycle component. The following illustrates how changing the pruning interval for Capture can affect the invocation of the Source Life Cycle component instance.



Increasing the Capture instance prune_interval parameter from 2 time units (for example, 300 seconds) to 3 time units (for example, 450 seconds) causes:

- Rows in the Capture work tables, which are eligible for removal, to remain longer in the work table, thus increasing the potential space requirements. Work tables will grow larger, but the system load and the risk of contingencies can be lower.
- Rows in the source tables, which can be removed based on the life cycle retention policy, to remain longer than expected in the source table.

In general, if the Capture prune_interval parameter is set to a value larger than the prune_interval parameter for the life cycle component, the Capture parameter setting takes precedence. If a Capture utility is not running or if its auto pruning feature is disabled, no source life cycle enforcement will be performed.

Configuring the Source Life Cycle component

Multiple life cycle component instances are being used in each source database (State and Runtime databases). Each instance, which is implemented by a trigger, enforces the retention policies as defined in the control table WBIRMADM.RMPRUNECTRL located on the source database for that data movement service. Life cycle retention policies are specified on a per table basis. Thus, one row in WBIRMADM.RMPRUNECTRL corresponds to one table requiring pruning.

TABLE_NAME	RETEN...	LAST_PRUNED	PRUNE_IN...	PRUNE_EN...	LOG...	ROW...
wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI	1440	Oct 11, 2005 4:40:...	5	1	0	0
wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE	1440	Oct 11, 2005 4:40:...	5	1	0	0

- **Identifying Source Life Cycle component instances**.

  To determine which triggers have been assigned to enforce retention policies for a given business measures model you must:

  – Identify the data movement service for which you want to modify the ETL configuration.

  – Inspect the WBIRMADM.RMMETADATA table in the State database (for the State to Runtime data movement service) or the Runtime database (for the Runtime to Historical data movement service), and look up the associated trigger names in column SRC_RM_PRUNE_TRG_NAME.

    Example: The query "SELECT OM_NAME, SRC_TAB_NAME, SERVICE_NAME, SRC_RM_PRUNE_TRG_NAME FROM WBIRMADM.RMMETADATA WHERE SERVICE_NAME='State to Runtime'" may result in the following:

OM_NAME	SRC_TAB_NAME	SERVICE_NAME	SRC_RM_PRUNE_TRG_NAME
STEW_S	wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	State to Runtime	WBIRMADM.MCPruneTrig_8
STEW_S	wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE	State to Runtime	WBIRMADM.MCPruneTrig_9

    In this example, two triggers (WBIRMADM.MCPruneTrig_8 and WBIRMADM.MCPruneTrig_9) are enforcing the life cycle retention policy for the business measures model STEW_S source tables in the State database. Because retention policies are defined by table and not by the life cycle component instance names, keep track of column SRC_TAB_NAME when planning to alter the life cycle enforcement behavior.

- **Modifying Source Life Cycle component instance configurations**.

  – Enabling and disabling life cycle component instances:

    Pruning can greatly affect the performance of your system. When pruning is enabled, it reduce the amount of information that transaction servers (State) and reporting servers (Runtime) have to contend with. It also adds a small additional load on those same tables during every invocation according to the parameters of the Life Cycle component. When pruning is disabled, the source tables will grow over time, which can also degrade performance.

    By default, source tables are automatically pruned according to their life cycle retention policy. To temporarily disable pruning, modify the corresponding WBIRMADM.RMPRUNECTRL entries: set column PRUNE_ENABLED to 1 to enable pruning, or, to disable pruning, set any other numeric value (preferably zero).

    Rows will be purged from source table wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI, but no rows will be purged from table wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE if the following configuration is used. The query: "SELECT TABLE_NAME, PRUNE_ENABLED FROM WBIRMADM.RMPRUNECTRL" may result in the following:

TABLE_NAME	PRUNE_ENABLED
wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	1
wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE	0

  – Changing the retention policy:

    Retention time policies can be changed for source tables located in the Runtime database only. For all tables located in the State database, a retention period of 0 is enforced, regardless of the settings in WBIRMADM.RMPRUNECTRL. A retention period is defined as the minimum

time that a row must be retained in a source table until it can be removed, if it meets two criteria. Of the two criteria, only one is customizable through the control table: the retention time specified in minutes. Any row that has been marked as ready for deletion and has remained in the source table for at least RETENTION_IN_MINUTES becomes eligible for removal.

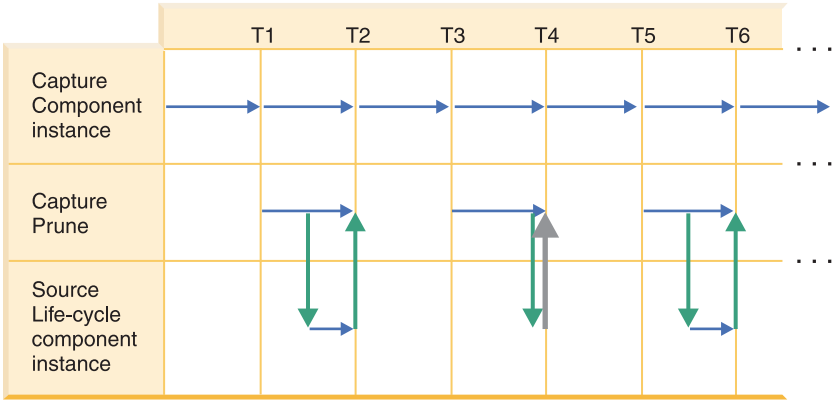
Using the default configuration for source tables in the Runtime database, rows that have been marked as ready for deletion by the server need to be kept for one day (1440 minutes) before they can be removed. The query: "SELECT TABLE_NAME, RETENTION_IN_MINUTES FROM WBIRMADM.RMPRUNECTRL" may result in the following:

TABLE_NAME	RETENTION_IN_MINUTES
wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI	1440
wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE	1440

Changes to the WBIRMADM.RMPRUNECTRL control-table entries will be picked up each time the Source Life Cycle component is invoked.

– Scheduling source data pruning:

  There is a dependency between the Capture work table pruning interval and the Source Life Cycle component invocation. An invocation will not result in execution if not enough time has elapsed between Source Life Cycle component instance invocations, as shown in the following figure.



Assuming that the Source Life Cycle component is scheduled to execute every 4 time units, but the Capture component is configured to prune its work tables every 2 time units, invocation at time T4 will not result in execution.

To change the default schedule, locate the appropriate entries in WBIRMADM.RMPRUNECTRL, and alter the column value for PRUNE_INTERVAL, which represents the minimum delay in minutes between executions.

TABLE_NAME	LAST_PRUNED	PRUNE_INTERVAL
wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	Oct 11, 2005 5:16:44 PM ...	5
wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE		5

Increasing the value results in less frequent executions (but the number of invocations stays the same). Each execution determines which source table rows are eligible for removal and removes them. Regularly monitor your

source databases to identify and eliminate potential performance problems that are a result of locks, resulting from these deletions.

Configuring the (Target) APPLY Component

An instance of an Apply component is a DB2 Apply replication utility. Changes that have been captured by Capture utilities are continuously applied to staging tables in the target database by default. The default utility configuration parameters should be sufficient for most environments and should not be changed.

- **Identifying Apply component instances**.

  Multiple Apply component instances (DB2 Apply utilities) are used to apply any data changes to internal staging tables associated with a business measures model. To determine which Apply utilities have been assigned to provide services for a business measures model:

  – Identify the data movement service for which you want to change the Apply utility configuration

  – Inspect the WBIRMADM.RMMETADATA metadata table in the Runtime database (for the State to Runtime data movement service) or in the Historical database (for the Runtime to Historical data movement service), and identify all Apply utility names (column TGT_RM_APP_SVR_NAME). The query: ″SELECT OM_NAME, SRC_TAB_NAME, SERVICE_NAME, TGT_RM_APP_SVR_NAME FROM WBIRMADM.RMMETADATA WHERE SERVICE_NAME='State to Runtime'″ may result in the following:

OM_NAME	SRC_TAB_NAME	SERVICE_NAME	TGT_RM_APP_SVR_NAME
STEW_S	wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	State to Runtime	APPLY_4
STEW_S	wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE	State to Runtime	APPLY_4

  In this example, any data changes for the business measures model STEW_S that have been captured in the State database will be applied to staging tables in the Runtime database by Apply utility APPLY_4.

  Each time the Apply component finishes processing all (committed) changes that have been recorded by the Capture utility, one or more ETL component instances and Target Life Cycle component instances are invoked.

Configuring the ETL Component

ETL components have been implemented in WebSphere Business Monitor as database stored procedures. These stored procedures always reside on the target database for any given data movement service. Therefore, all ETL stored procedures assigned to the State to Runtime data movement service are located in the Runtime database, and ETL stored procedures assigned to the Runtime to Historical data movement service reside in the Historical database.

- **Identifying ETL component instances**.

  Multiple ETL component instances are set to process any data that has been added to the internal staging tables associated with a business measures model. To determine which stored procedures have been assigned to provide services for a given business measures model:

  – Identify the data movement service for which you want to modify the ETL configuration.

  – Inspect the WBIRMADM.RMMETADATA metadata table in the Runtime database (for the State to Runtime data movement service) or in the Historical database (for the Runtime to Historical data movement service), and identify all ETL stored procedure names (column TGT_RM_SPETL_NAME). The

following query: "SELECT OM_NAME, SRC_TAB_NAME, TGT_TAB_NAME, SERVICE_NAME, TGT_RM_SPETL_NAME FROM WBIRMADM.RMMETADATA WHERE SERVICE_NAME='State to Runtime'" may result in the following:

OM_NAME	SRC_TAB_NAME	TGT_TAB_NAME	SERVICE_NAME	TGT_RM_SPETL_NAME
STEW_S	wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI	State to Runtime	WBIRMADM.WBIRMSP_10
STEW_S	wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE	wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE	State to Runtime	WBIRMADM.WBIRMSP_14

In this example, any data changes for the business measures model STEW_S that have been captured in the State database and applied to staging tables in the Runtime database, will be processed by stored procedures named WBIRMADM.WBIRMSP_10 and WBIRMADM.WBIRMSP_14. Successfully processed data will be stored in the target tables (identified by column TGT_TAB_NAME) in the Runtime database.

- **Modifying ETL component instance configurations**.

  ETL component instance configurations are stored in the WBIRMADM.RMCONTROL control table. Instances that have been assigned to the State to Runtime data movement service maintain their configuration in the Runtime database; all others, in the Historical database. Any changes that are made to a configuration will be picked up by the stored procedures upon the next startup. Three options can be configured through the control table:

  – Minimum elapsed time between two ETL executions (ETLSCHEDMETHOD, ETL_0_MINUTES)

  – Granularity of logging output (LOGLEVEL)

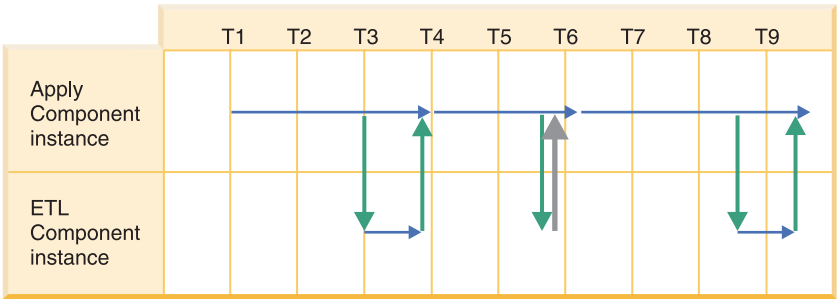
  – Transaction durations (COMMITINTERVAL)

  Each row in this table corresponds to one ETL component instance, which corresponds to exactly one target table that needs to be populated. The following example configuration illustrates how changes to the configuration affect the instances behavior.

TARGETTABLE	TGT_RM_SPETL_NAME	ETLSCHEDMETHOD	ETL_0_MINUTES	LOGLEVEL	COMMITINTERVAL
wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI ...	WBIRMADM.WBIRMSP_10	1	5	0	1000
wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE ...	WBIRMADM.WBIRMSP_14	1	5	0	1000

- **Changing the ETL schedule**.

  ETL component instances are invoked each time an Apply component instance has finished processing a subscription set. Upon invocation, an ETL instance checks its schedule and either starts processing or returns control immediately to the Apply component instance. It uses information stored in control table WBIRMADM.RMCONTROL to determine whether it needs to execute. The figure below shows the differences between invocation and execution: the first and third time, the ETL component instance is being executed, according to schedule. The second invocation occurred outside the schedule and does not result in any processing activity.



Various factors determine how often ETL component instances should run in the State to Runtime data movement service and the Runtime to Historical Data movement service:

- Availability: How soon should data be accessible in the target tables. Choosing a lower interval causes data to be made available earlier, but it also increases system load.
- Data volume: The replication utilities are continuously (or as configured) feeding data into the staging tables, whether or not it is being processed by ETL component instances. The more database resources are used, the more data needs to be processed. Processing data more often can reduce the maximum resource usage.
- Processing time: ETL processing takes less time for data in the Runtime database than processing of data in the Historical database. Plan the schedules accordingly. Using a small delay between executions will not give better performance if an execution lasts longer than the scheduled delay. For example, if it takes an ETL component instance 60 seconds to finish processing, a schedule interval of 30 seconds becomes effectively a schedule interval of at least 60 seconds because ETL component instances execute sequentially.

Two scheduling modes are currently supported:

- Flexible schedule:

  The ETL instance executes if at least ETL_0_MINUTES have passed since its last execution (LASTUPDATED). For example, assuming the control table contains the following information.

TGT_RM_SPETL_NAME	TARGETTABLE	LASTUPDATED	ETL_0_MINUTES
WBIRMADM.WBIRMSP_10	wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI ...	Oct 11, 2005 5:20:20 PM ...	60
WBIRMADM.WBIRMSP_14	wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE ...	Oct 11, 2005 5:20:20 PM ...	60

  The stored procedure WBIRMADM.WBIRMSP_10 would execute no earlier than October 11, 2005, 6:20.20 p.m. (October 11, 2005, 5:20.20 p.m.+ 60 minutes). Schedules can slip if a stored procedure is invoked later than October 11, 2005, 6:20.20 p.m. Assume that the current time is 7 p.m. and the stored procedure did not execute as expected at 6:20 p.m. The stored procedure is invoked and executes (about 40 minutes late). It will not execute again until at least 7 p.m.+ 60 minutes = 8 p.m. The effective schedule has slipped because ETL procedures were scheduled to run every 60 minutes at 20 minutes past the hour, but now run every 60 minutes at the full hour. If required, you can reset the schedule by changing the timestamp value in column LASTUPDATED.

  Use this scheduling mechanism if no fixed execution-time window is necessary. To enable this form of scheduling, set column

ETLSCHEDMETHOD in WBIRMADM.RMCONTROL to 0 for all stored procedures that have been assigned to the same business measures group:

TGT_RM_SPETL_NAME	TARGETTABLE	ETLSCHEDMETHOD
WBIRMADM.WBIRMSP_10	wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI ...	0
WBIRMADM.WBIRMSP_14	wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE ...	0

– Fixed schedule:

This is the default schedule for all ETL components. The ETL component instances execute if the current time is past NEXTSTARTTIME. To avoid slippage, the next scheduled execution time is calculated, based on the current time and the previously scheduled execution time, each time a stored procedure executes. The following example illustrates this schedule:

TGT_RM_SPETL_NAME	TARGETTABLE	LASTUPDATED	ETL_0_MINUTES	NEXTSTARTTIME
WBIRMADM.WBIRMSP_10	wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI ...	Oct 11, 2005 5:20:20 PM ...	60	Oct 11, 2005 6:20:20 PM ...
WBIRMADM.WBIRMSP_14	wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE ...	Oct 11, 2005 5:20:20 PM ...	60	Oct 11, 2005 6:20:20 PM ...

Assuming the current time is 7 p.m. and the stored procedures have not been executed as expected at 6:20 p.m. The stored procedures are executed because it is past NEXTSTARTTIME (6:20 p.m.) on the same day. The next execution will be scheduled for 7:20 p.m., in accordance with the original schedule, not 8 p.m.as in the case of a flexible schedule. If stored procedures have to start executing within a predefined time window, use this scheduling method. To enable this form of scheduling, set column ETLSCHEDMETHOD in WBIRMADM.RMCONTROL to 1 for all stored procedures that have been assigned to the same business measures group:

TGT_RM_SPETL_NAME	TARGETTABLE	ETLSCHEDMETHOD
WBIRMADM.WBIRMSP_10	wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI ...	1
WBIRMADM.WBIRMSP_14	wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE ...	1

It is strongly advised that the same scheduling method be used for ETL component instances belonging to the same business measures model because of interdependencies between those instances. This is especially important in the Historical database and for schedules with long intervals (several hours or more). Setting ETLSCHEDMETHOD to a value other than 0 or 1 will disable the ETL component instance.

- **Changing the logging level**.

  Two levels of logging are supported by the stored procedures: minimum (0) and maximum (1). To modify the default setting of minimum, change the value of column LOGLEVEL in WBIRMADM.CONTROL for the stored procedures (TGT_RM_SPETL_NAME), whose logging level needs to be changed. All logging output is appended to WBIRMADM.RMLOG. The two example stored procedures WBIRMADM.WBIRMSP_10 and WBIRMADM.WBIRMSP_14 both perform minimal logging:

ENTRYSTMP	NAME	OPERATION	ISTRACEENTRY	ID
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_14	SP_START	0	
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_14	DEL_TEMP	0	
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_14	INS_TEMP	0	
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_14	FETCH_TARGET_...	0	
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_14	SP_END	0	
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_10	SP_START	0	
Oct 11, 2005 4:40:20 PM ...	WBIRMADM.WBIRMSP_10	DEL_TEMP	0	

Because the log table is not automatically pruned, it needs to be monitored regularly by the DBA. Keep logging output to a minimum unless instructed otherwise.

- **Changing transaction durations**.

  Data that has been successfully processed by the stored procedure is written to the target tables immediately. However, depending on the commit interval setting (column COMMITINTERVAL in WBIRMADM.RMCONTROL), any updates to the target table are not made permanent until the specified number of rows has been processed or until no more rows are left to be processed. Increasing the value of COMMITINTERVAL (for example, to 1500) will force the stored procedure to process more data before committing any changes. Locks on the target table will be held longer and may have a negative impact on other components that are trying to access the same table. Decreasing the duration (for example, to 500) lowers the number of rows that need to be processed before they are made available in the target table and releases locks earlier.

TARGETTABLE	TGT_RM_SPETL_NAME	ETL_0_MINUTES	LOGLEVEL	COMMITINTERVAL
wbi.CTR_TQ4MUFT42JOT5F6R3KSDQDE2UI ...	WBIRMADM.WBIRMSP_10	5	0	1500
wbi.AIR_BVSOYAP1DRWFD5HNQJR5HFQQQE...	WBIRMADM.WBIRMSP_14	5	0	500

Configuring the Target Life Cycle component.

ETL work tables continuously grow as long as new or updated data is applied by Apply component instances. One Target life Cycle component instance, implemented by a stored procedure, is assigned to one work table in each target (Runtime and Historical) database. Each instance enforces the internal retention policies as defined in control table WBIRMADM.RMPRUNECTRL. As in the source tables, life cycle retention policies for ETL work tables are specified on a per table basis. Thus, one row in WBIRMADM.RMPRUNECTRL corresponds to one table that requires pruning.

- **Identifying Target Life Cycle component instances**.

  To determine which stored procedures have been assigned to enforce ETL work table retention policies for a given Business Measures model:

  – Identify the data movement service for which you want to modify the ETL configuration.

  – Inspect the WBIRMADM.RMMETADATA table in the Runtime database (for the State to Runtime data movement service) or Historical database (for the Runtime to Historical data movement service), and look up the associated stored procedure names in column TGT_RM_APP_PRUNE_SP_NAME in the following table.

OM_NAME	SRC_TAB_NAME	TGT_RM_APP_STG_TAB_NAME	TGT_RM_APP_PRUNE_SP_NAME	SERVICE_NAME
STEW_S	wbi.CTX_TQ4MUFT42JOT5F6R3KSDQDE2UI	APP.CCD_6	WBIRMADM.WBIRMSP_P_13	State to Runtime
STEW_S	wbi.AI_BVSOYAP1DRWFD5HNQJR5HFQQQE	APP.CCD_7	WBIRMADM.WBIRMSP_P_17	State to Runtime

In this example, two stored procedures (WBIRMADM.WBIRMSP_P_13 and WBIRMADM.WBIRMSP_P_17) are enforcing the life cycle retention policy for the business measures model STEW_S ETL work tables in the Runtime database. Because retention policies are defined by table and not by life cycle component instance names, keep track of column TGT_RM_APP_STG_TAB_NAME when planning to alter the life cycle enforcement behavior.

- **Modifying the configurations of a Target Life Cycle component instance**.

  The default configurations should be appropriate for most deployments but can be fine tuned as described in the following:

  – Enabling and disabling Target Life Cycle component instances.

    By default, ETL work tables are automatically pruned according to their life cycle retention policy. To temporarily disable pruning, modify the corresponding WBIRMADM.RMPRUNECTRL entries: set column PRUNE_ENABLED to 1 to enable pruning, or, to disable pruning, set any other numeric value (preferably zero). Both ETL work tables are automatically pruned, if control table WBIRMADM.RMPRUNECTRL contains the following entries in the Runtime database:

TABLE_NAME	PRUNE_ENABLED
APP.CCD_6	1
APP.CCD_7	1

    Before disabling any of the Target Life Cycle component instances, make sure enough space is available in the associated table space containers. Each time the Monitor server updates any rows in the source tables, one row is added to the ETL work tables. Thus one row in a source table can be temporarily represented by multiple rows in the work tables, making work tables grow much faster than source tables. Changes to WBIRMADM.RMPRUNECTRL will be picked up the next time a Target Life Cycle component instance is invoked.

  – Changing the retention policy.

    All rows that have been successfully processed by ETL component instances can be removed from the work tables. The default retention period for both the Runtime and Historical databases, is set to zero minutes:

TABLE_NAME	RETENTION_IN_MINUTES
APP.CCD_6	0
APP.CCD_7	0

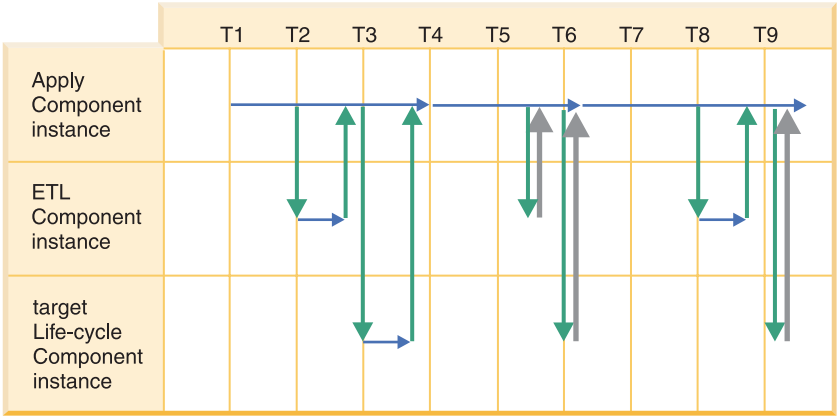
    All eligible rows will be removed the next time a Target Life Cycle component instance is invoked. Setting the retention period to zero does not guarantee immediate removal because the schedule determines when the life cycle component is executed.

    The user can keep the data in this table by changing column RETENTION_IN_MINUTES in WBIRMADM.RMPRUNECTRL to another duration in minutes.

  – Scheduling ETL work table pruning.

    The concept behind Target Life Cycle component scheduling is very similar to ETL component scheduling. Upon completion of an Apply cycle and the

completion of all associated ETL component instances, Target Life Cycle component instances are invoked sequentially.



An invocation results in execution, if work table pruning is enabled and if it is scheduled to run. In the above example, the Target Life Cycle component instance is only executed once, at time T3. There are several reasons why execution did not occur at time T6 and T9:

- The Target Life Cycle component instance configuration was changed between T4 and T6, and pruning was disabled.
- The elapsed time between T3 and T9 is lower than the interval specified for this Target Life Cycle component instance.

Note: In this example, the ETL component instance and the Target Life Cycle component instance schedules are different (assuming that pruning was not disabled).

In general, either use the same schedule for all corresponding instances or use a schedule with longer cycles for the life cycle instances. The reason is that no data can be pruned unless it has already been successfully processed by the ETL component instances. To change the default schedule, locate the appropriate entries in WBIRMADM.RMPRUNECTRL. and alter the column value for PRUNE_INTERVAL, which represents the minimum delay in minutes between executions:

TABLE_NAME	LAST_PRUNED	PRUNE_INTERVAL
APP.CCD_6	Oct 11, 2005 4:40:20 PM ...	1440
APP.CCD_7	Oct 11, 2005 4:40:20 PM ...	1440

Increasing the pruning interval results in less frequent executions, but the number of invocations stays the same. Each execution determines which work table rows are eligible for removal and removes them. Work tables APP.CCD_6 and APP.CCD_7 will be pruned roughly once every day (1440 minutes) if the above configuration is used. Pruning was performed the last time at 4:40 p.m. on October 11th and will be executed again on October 12th around 4:40 p.m.

Summary of the configuration parameters of data movement services

The following table summarizes the most commonly used parameters provided for each of the data movement services components. For more information about configuration parameters, refer to DB2 Replication guide.

Component	Parameter name	Default values	Valid values	Parameter location
Capture	autoprune	Y		
Capture	prune_interval (seconds)	300		
Source Life Cycle	PRUNE_ENABLED	1	0 - Disabled 1 - Enabled	Data movement service Source DB: WBIRMADM.RMPRUNECTRL
Source Life Cycle	RETENTION_IN_MINUTES	0 - State to Runtime 1440 - Runtime to Historical	0 to DB2 limit for BIGINT	Data movement service Source DB: WBIRMADM.RMPRUNECTRL
Source Life Cycle	PRUNE_INTERVAL (minutes)	5	0 to DB2 limit for BIGINT	Data movement service Source DB: WBIRMADM.RMPRUNECTRL
ETL	ETLSCHEDMETHOD	1	0 - Flexible scheduling 1 - Strict interval scheduling Other - Disables ETL	Data movement service Target DB: WBIRMADM.RMCONTROL
ETL	ETL_0_MINUTES	5 - State to Runtime 1440 - Runtime to Historical	0 to DB2 limit for INTEGER	Data movement service Target DB: WBIRMADM.RMCONTROL
ETL	LOGLEVEL	0	0 - For normal logging 1 - For trace logging	Data movement service Target DB: WBIRMADM.RMCONTROL
ETL	COMMITINTERVAL (number of records.)	1000	0 - Disable commits until the end 1 - Commit every record. n - DB2 Limit for BIGINT	Data movement service Target DB: WBIRMADM.RMCONTROL
Target Life Cycle	PRUNE_ENABLED	1	0 - Disabled 1 - Enabled	Data movement service Target DB: WBIRMADM.RMPRUNECTRL
Target Life Cycle	RETENTION_IN_MINUTES	0	0 to DB2 limit for BIGINT	Data movement service Target DB: WBIRMADM.RMPRUNECTRL
Target Life Cycle	PRUNE_INTERVAL (minutes)	1440	0 to DB2 limit for BIGINT	Data movement service Target DB: WBIRMADM.RMPRUNECTRL

Note: IBM® reserves the right to make changes to the database tables and columns referenced above. As such, some tables and columns may be altered, removed, or added from release to release. Any reliance on the content or structure referenced in the information from release to release is done at the consumer's own risk. IBM will document any such changes as they occur.

Change management and artifacts generation

The business measures model includes many events and processes definitions. Based on these definitions, the Schema Generator generates the corresponding artifacts needed for creating the database tables, Cube Views definitions, and replication scripts. Changes in the business measures model induces changes in the generated artifacts.

If such a change occurs, you need to rerun the Schema Generator to generate the new business measures model scripts. This activity is called change management.

Change management is required in the following cases:

- A new process is added, resulting in the addition of a new table.
- A new metric is added, which is not part of a dimension, or is created in a new dimension, and a new column is added to the appropriate process tables.
- The length of a metric of type string is changed, resulting in a change in the corresponding column length.

Any changes in the business measures model require you to repeat the following steps:

- Import the changed business measures model into the Schema Generator in the WebSphere Business Monitor administrative console to generate the corresponding artifacts.
- Run the newly generated Database Definition Language (DDL) scripts to update the database with the changes.
- Deploy the replication scripts to synchronize the State, Runtime, and Historical databases after the changes.
- Deploy the newly generated Cube Views definitions.
- Deploy the business measures model into the WebSphere Business Monitor administrative console.

The Schema Generator examines the previous version of the business measures model and the new version. If the new model has not been deployed or does not exist in the repository, the Schema Generator generates the artifacts that correspond to the new model. If a previous version of the new model exists, the Schema Generator examines the differences between the deployed model version and the new model version. If changes are found, the appropriate scripts are regenerated to modify the databases, based on these differences. For more information on creating a new model from an existing one, refer to WebSphere Business Modeler documentation.

Some changes in the business measures model are restricted because of limitations in altering the database tables of the existing business measures model. If the following changes have occurred, regenerate the entire model and redeploy it as a new one. A full new set of artifacts is generated and deployed. The changes are:

- Changing the **Usage in WebSphere Business Monitor** attribute of a business measure, for example, changing a metric from a **Active data about running process** value to a **Quantitative data in dimensional analysis** value.
- Changing the dimension group to which a metric belongs.
- Changing the status of **isPartOfDimensionKey** check box on a dimension metric in the Business Measures editor.
- Changing the data type of the metric. Changes in the metric data type are handled by deleting a current metric and creating a new one.

- Deleting a metric that is part of a dimension key.
- Creating a new metric as part of a dimension key for an existing dimension.
- Changing a timer between accumulating and non-accumulating types.
- Changing the process model or the activities.

Note: Deleting an entire process does not require model regeneration even though that it may result in the deletion of a metric. Only the changes could be generated.

According to the changes in the business measures model, there are three deployment scenarios:

- New model deployment
- New version model deployment
- Different model deployment

A new process is added

When you add a new process to the business measures model, a new table is added to the State, Runtime, and Historical databases.

After you add the new process, you use the Schema Generator to generate the scripts needed to alter the created databases (State, Runtime and Historical databases) and replication settings.

Note: It is important to have gone through the planning exercises with this change. For more information about database planning, refer to "Preparation of database artifacts deployment" on page 7.

To synchronize replication among databases, update the replication configuration with the new tables that correspond to the newly added processes. Deploy these scripts to add the new tables in WebSphere Business Monitor databases and make the appropriate changes in the replication settings to reflect the changes made in the database tables.

When a new process is added, you need to the following:

- Back up all databases before you deploy any new or updated business measures model.
- Use the Schema Generator to generate the Database Definition Language (DDL) scripts that are deployed to add the new tables and alter the existing databases.
- Deploy the new replication scripts on the DB2 command window to enable replication of the new processes tables.
- Re-import the cube model into DB2 Cube Views and ALPHABLOX® so that the cube model reflects any new cubes that have been created in the Historical database.

An existing Business Measures group has a new column

When you add a new business measure (metric) to an existing Business Measures group, a new column is added in some instances tables in the State, Runtime, and Historical databases.

The tables, which are updated with the new columns, are selected according to the **Usage in WebSphere Business Monitor** attribute of the metric being added. The affected databases are shown in the following table:

Metric usage in WebSphere Business Monitor databases

Usage in WebSphere Business Monitor	State database	Runtime database	Historical database
Temporary calculation	Yes	No	No
Active data about running processes	Yes	Yes	No
Quantitative data in dimensional analysis	Yes	Yes	Yes
Aggregation group in dimensional analysis	Yes	Yes	Yes

After you add a new business measure (metric) in the Business Measures editor, you use the Schema Generator to generate the scripts needed to alter the created databases (State, Runtime and Historical databases) and replication settings. To alter the databases, deploy the Database Definition Language (DDL) scripts on the DB2 command window to add the new columns to the database tables.

To synchronize replication among databases, update the DB2 Replication Center with the new columns that correspond to the newly added metric by deploying replication scripts on the DB2 command window.

When a new metric is added to a process, you need to do the following:

- Backup all databases before you deploy any new or updated business measures model.
- Stop the monitoring service on that process.
- Stop the replication service for that process.
- Use the Schema Generator to generate the DDL scripts that are deployed to add new columns and alter the existing databases.
- Deploy the generated replication scripts to reflect the changes in the topology of the databases.
- Re-import the cube model into DB2 Cube Views and ALPHABLOX so that the cube reflects any new dimensions that have been created in the Historical database

Database maintenance

WebSphere Business Monitor databases require regular maintenance. Some DB2 tools can help you in enhancing the use and performance of the databases.

The recommended tools are:

- Configuration Advisor.
- DB2 Web Health Center, which assists database administrators (DBAs) by alerting them to potential problems and providing recommendations to resolve problems. DBAs can monitor an instance remotely using the Web Health Center and view alert details and make recommendations.
- REORG and REORGCHK commands. REORG eliminates overflow rows and reclaims space from deleted rows of tables and indexes. This tool is useful if there have been a large number of deletions, updates, or inserts. REORGCHK

updates the statistics that are used by the DB2 optimizer tool. This tool is useful when the statistical information of the database tables are not up to date due to database updates.

- RUNSTATS command, It gathers statistics about databases objects. These statistics can be used during data retrieval to choose the path in accessing data. Therefore the DB2 will have the information it needs to choose the most efficient access path. Databases performance will be poor until an administrator runs the DBA RUNSTATS command against all of the tables in all of the databases after some data has been allowed to accumulate in each of the databases. Example:

  ```
  RUNSTATS ON TABLE tablename WITH DISTRIBUTION AND DETAILED INDEXES ALL
  ```

  Note: The *tablename* must be fully qualified with the schema name.
  After the running the command. Run DB2 rebind command:

  ```
  db2rbind <database_alias> -l logfile all
  ```

  The performance affects Monitor Server event processing performance as well as Replication performance. In addition the default Pruning interval set in replication's RMPRUNECTL table should be defaulted to 0 instead of the current 1440 seconds (or 24 hours).

For more information on DB2 maintenance utilities, refer to IBM DB2 documentation.

Backing up databases and recovering them from deployment errors is also part of database maintenance.

Database backup

You should back up Repository, State, Runtime and Historical databases before you run any new WebSphere Business Monitor Database Definition Language (DDL) scripts, whether they deploy a new business measures model or update an existing one.

Backup ensures a safe rollback point if something fails. If you are not concerned with recent data collection, you can roll the database back to a prior state that does not contain tables for a recently deployed business measures model.

For more information about database backup and restore, refer to DB2 data recovery section.

Recovery after deployment errors

If you encounter an error while deploying replication artifacts. You have to undo the actions performed within the deployment of a single business measures model to undo the changes.

All deployments are done in several stages, the following are the typical scenario:

1. DDL deployment
   a. Deploy state.ddl.
   b. Deploy runtime.ddl.
   c. Deploy datamart.ddl.
2. Data movement services deployment
   a. Deploy State_to_Runtime_setup_source.
   b. Deploy State_to_Runtime_setup_target.

c. Deploy Runtime_to_historical_setup_source.
d. Deploy Runtime_to_Historical_setup_target.

You must identify at what point the failure has occurred to determine how to take action. For example, if the state.ddl fails, then it is simply a matter of rolling back the transaction to get back to the original state. However, if the datamart.dll fails, then rolling back the datamart.ddl will only get the system back to the point after the runtime.ddl executed successfully. Failures in the middle of the data movement services deployment are the most difficult to recover from, however it is not impossible. First deployments are the easiest to recover from, deployments of new models are next, and finally deployments of change models represents the hardest recovery paths.

To recover from the replication scripts deployment errors, you go through the following stages: identifying, backing up, restoring or removing, and redeploying:

Identifying

- Identify the errors that occurred and determine whether IBM Support should be called.
- Identify the business measures model being deployed when the error occurred.
- Identify the Schema Generator tables being created or altered when the error occurred.
- Identify the Schema Generator artifacts being created or altered when the error occurred.
- Identify the last valid version of the business measures model in the Repository database.
- Identify, if a change management deployment, the location of the artifacts which were deployed for previous versions of the model. This will give the database structures, there descriptions and relationships to each other. This may be important in case data needs to be backed up and later restored.
- Identify the location of the current artifacts and deployment log files. These will be important for problem determination and potentially to provide to IBM Support.
- Identify, if a change management deployment, whether there is data that exists in any of the CCD tables that has not been processed yet. You can use the *WBIRMADM.RMMETADATA* table (available on the Runtime and Historical databases) to determine the associated CCD tables (*TGT_RM_APP_STG_TAB_NAME*) with the business measures model project name (*OM_NAME*) that was being deployed. Any rows that are marked with an **I** or a **U** may not have been processed and should be backed up. The column *SERVICE_NAME* contains the location of the CCD table and the target table, the name after the word *to* indicates this. You should keep track of the relationship to the *TGT_TAB_NAME* in case you decide to completely Remove all the artifacts and generate a completely new set. This is because Schema Generator may not generate the same names for the CCD tables and you will need to restore this data into the new CCD tables after deployment is successful.

Backing up

- Determine, based on the identification stage, whether any data needs to be backed up. Because other business measures models may be running at the same time during the deployment, you may need to back up the database objects associated with the other business measures models.

- You may have to back up the CCD (Consistent-Change Data) tables. The Schema Generator may still have some information in these tables that does not exist in the source or target database tables yet.

  Note: Completed events may only exist in the Historical database.

Restoring or removing

- Determine if it is easier to restore the previous database or to remove artifacts manually.
- **Restoring** Restoring from a backed up version may be a good choice when no other business measures models exist, or when other business measures models have had no activity. Restore the previous database set, and for each database, rebind any applications that need to be rebound, and reregister all the Java-based stored procedures and user-defined functions.

  Note:

  – For more information about database backup and restore, refer to the Data Recovery section in the DB2 documentation.
  – For completed deployed models, the *WBIRMADM.RMMETADATA* table provides information on what not to remove. However, while deployments, in order to determine some of the artifacts and relationships, it may be necessary to examine the deployment logs to determine what can be removed safely.

- **Removing**
  – Replication: Historical database and Runtime database
    - Stop all Capture servers associated with that business measures model. (The Capture servers run on the Runtime and State databases.)
    - Stop all Apply Servers associated with the business measures model.
    - Remove all ETL stored procedures for the business measures model.
    - Remove all ETL staging tables that are used for the business measures model.
    - Remove all ETL control information from the WBIRMADM.RMCONTROL table in the corresponding target database for that business measures model.
    - Remove all ETL pruning stored procedures and triggers that are used for the business measures model.
    - Remove all tables listed in the *WBIRMADM.RMMETADATA* table column *TGT_RM_APP_STG_TAB_NAME* that have the suffix *_BKUP* and *_M* and also have a corresponding *SERVICE_NAME* of *Runtime_to_Historical* for Historical and *State_to_Runtime* for Runtime for that business measures model. Leave the table listed in *TGT_RM_APP_STG_TAB_NAME* as this will be removed in a later step.
    - Using the DB2 Replication Center, Remove all Apply subscription set members serving that business measures model.
      • If the Apply subscription set is empty, remove the subscription set.
      • If the Apply server has no subscription sets, remove the Apply server.
    - Remove all metadata entries associated with the business measures model from the WBIRMADM.RMMETADATA table. You will also need to remove the same entries from the *WBIRMADM.RMMETADATA* table in the Runtime database if processing the Historical database and for the State database if processing the Runtime database. You must only remove the rows for the business measures model and that are in the

Runtime_to_Historical service name when processing the Historical database and the *State_to_Runtime* for the Runtime database.

– Replication: Runtime database and State database
  - Stop all Capture servers serving the business measures model.
  - Remove all the triggers associated with the Capture CD tables that are associated with the business measures model.
  - Remove pruning control information from the WBIRMADM.RMPRUNECTRL table for pruning triggers that are used for the business measures model.
  - Using the DB2 Replication Center, Remove all of the subscriptions for all of the tables associated with the business measures model.
  - Remove all metadata entries associated with the business measures model from the WBIRMADM.RMMETADATA table.
– Database schema: Generally, an error during schema generation is rolled back before the altered model is deployed. The current set of replication artifacts is not affected.

Redeploying

When you have removed all the artifacts supporting a business measures model, you can run the Schema Generator again with the **Ignore Older Deployments** option selected. If the generated schema succeeds, do not deploy the Database Definition Language (DDL) scripts; deploy the replication scripts again.

Creating and configuring databases

Creating and configuring WebSphere Business Monitor databases is a critical phase in the installation process.

The WebSphere Business Monitor has four databases:

- Repository
- State
- Runtime
- Historical

At WebSphere Business Monitor installation, the databases can only be created using the Launchpad. If the databases are dropped after installation, you can re-create them using the Launchpad or manually. Creating the databases includes creating the static tables, table spaces, and indexes and setting the appropriate database configurations. The State, Runtime, and Historical databases contain both static and dynamic tables; the Repository database contains static tables only. The scripts required to create the databases are saved in *<monitor_installation_dir\install\mondb*.

To create the databases manually, perform the following steps in sequence:

1. **On Windows system:**
   a. Go to **Start -> Programs -> IBM DB2 -> Command Line Tools -> Command Window**.
   b. To create the Repository database, run the script: db2CreateRepository.bat <%RepositoryDatabaseName%> <%DB2userID%> <DB2Password> Create_Repository.sql <%Install_Directory%>.

c. To create the State database, run the script: db2CreateState.bat createStateDB.ddl <%Install_Directory%>.
d. To create the Runtime database, run the script: db2CreateRuntime.bat createRuntimeDB.ddl <%Install_Directory%>.
e. To create the Historical database, run the script: db2CreateHistorical.bat createDatamartDB.ddl <%Install_Directory%>.

Note: *<%Install_Directory%>* is "C:\IBM\Websphere\Monitor" by default.

2. **On AIX® system:**
   a. Login as the DB2 instance user.
   b. To create the Repository database, run the script: db2CreateRepository.sh <%RepositoryDatabaseName%> <%DB2userID%> <DB2Password> <%PathToDDL%>/Create_Repository.sql <%Install_Directory%>.
   c. To create the State database, run the script: db2CreateState.sh <%PathToDDL%>/createStateDB.ddl <%Install_Directory%> <%DB2UserID%>.
   d. To create the Runtime database, run the script: db2CreateRuntime.sh <%PathToDDL%>/createRuntimeDB.ddl <%Install_Directory%> <%DB2UserID%>.
   e. To create the Historical database, run the script: db2CreateHistorical.sh <%PathToDDL%>/createDatamartDB.ddl <%Install_Directory%> <%DB2UserID%>.

   Note:
   - If you installed to an alternate directory, you will have to substitute the appropriate paths based on your installation directory.
   - *<%PathToDDL%>* is /opt/IBM/Websphere/Monitor/Install/mondbby default.
   - *<%Install_Directory%>* is /opt/IBM/Websphere/Monitor by default.
   - *<%DB2UserID%>* is *db2inst1* by default.

Managing databases at run time

Managing WebSphere Business Monitor databases at runtime covers the deployment of the generated artifacts from the Schema Generator in the WebSphere Business Monitor administrative console. The deployment of these artifacts is repeated each time a new or a changed business measures model is imported.

Managing the databases at runtime consists of completing the following tasks.

Creating dynamic database tables

The dynamic database tables correspond to a specific business measures model. The required scripts for creating these tables are generated from the Schema Generator.

You must run the Schema Generator in the WebSphere Business Monitor administrative console to generate the required scripts for creating the dynamic database tables.

The scripts create tables and indexes and set the configuration parameters for each dynamic table in the State, Runtime, and Historical databases. The location of the

generated scripts is specified during the configuration of the Schema Generator in the WebSphere Business Monitor administrative console.

State database

You follow these steps to create dynamic database tables in the State database. The scripts are stored in a user-defined location. This location is defined during the setting of the Schema Generator configuration in the WebSphere Business Monitor administrative console.

The Database Definition Language (DDL) script needed for creating the dynamic database tables in the State database is stored in the **state.ddl** file on the root directory. To deploy the scripts, complete the following steps:

1. Open the DB2 **Command Window**. On UNIX, If the shell environment is configured, you can invoke the DB2 command line processor.
2. Change the path to the location of the script file.
3. Back up the State database before you deploy a new business measures model.
4. Run the command ***db2 terminate***. This ensures that any previous background processes which may be using a different code page value will not be used and that a new background process will be used to process this request.
5. Set the ***DB2CODEPAGE*** environment variable to 1208. The DB2 command line processor will by default interpret any character data using the current code page. The generated *state.ddl* however contains UTF-8 characters that will be corrupted unless the DB2CODEPAGE environment variable is set to 1208.
   a. On the UNIX operating system.
      - Using *sh*, *ksh*, *bash* type shells, run the command ***export DB2CODEPAGE=1208***.
      - Using *csh*, *tsch* type shells, run the command ***setenv DB2CODEPAGE 1208***.

   b. On the Windows operating system, run the command ***set DB2CODEPAGE=1208***.
6. Connect to the State database by running the command ***db2 connect to*** **<State_DB_Name>**.
7. Run the command ***db2 +c –stvf state.ddl > state.log***. This runs the script and saves a log file that records the transactions for troubleshooting purposes. Check the log file for any errors before you decide to commit or rollback. If a rollback is required, run the command ***db2 rollback*** to undo the actions. If no errors occurred, run the command ***db2 commit*** to commit the changes.
8. Disconnect from the State database after running the script with the command ***db2 disconnect*** **<State_DB_Name>**.
9. Run the command ***db2 terminate*** to terminate the background processes.

Runtime database

You follow these steps to create dynamic database tables in the Runtime database. The scripts are stored in a user-defined location. This location is defined during the setting of the Schema Generator configuration in the WebSphere Business Monitor administrative console.

The Database Definition Language (DDL) scripts needed for creating the running database tables in the Runtime database are stored in **runtime.ddl** file in the root directory. To deploy the scripts, complete the following steps:

1. Open the DB2 **Command Window**. On UNIX, If the shell environment is configured, you can invoke the DB2 command line processor.

2. Change the path to the location of the script file.
3. Back up the Runtime database before you deploy a new business measures model.
4. Run the command ***db2 terminate***. This ensures that any previous background processes which may be using a different code page value will not be used and that a new background process will be used to process this request.
5. Set the ***DB2CODEPAGE*** environment variable to 1208. The DB2 command line processor will by default interpret any character data using the current code page. The generated *runtime.ddl* however contains UTF-8 characters that will be corrupted unless the DB2CODEPAGE environment variable is set to 1208.
   a. On the UNIX operating system.
      - Using *sh*, *ksh*, *bash* type shells, run the command ***export DB2CODEPAGE=1208***.
      - Using *csh*, *tsch* type shells, run the command ***setenv DB2CODEPAGE 1208***.

   b. On the Windows operating system, run the command ***set DB2CODEPAGE=1208***.
6. Connect to the Runtime database by running the command ***db2 connect to*** **<Runtime_DB_Name>**.
7. Run the command ***db2 +c -stvf runtime.ddl > runtime.log***. This runs the script and saves a log file that records the transactions for troubleshooting purposes. Check the log file for any errors before you commit or rollback. If a rollback is required, run the command ***db2 rollback*** to undo the actions. If no errors occurred, run the command ***db2 commit*** to commit the changes.
8. Disconnect from the Runtime database after running the script by running the command ***db2 disconnect*** **<Runtime_DB_Name>**.
9. Run the command ***db2 terminate*** to terminate the background processes.

Historical database

You follow these steps to create the dynamic database tables in the Historical database. The scripts are stored in a user-defined location. This location is defined during the setting of the Schema Generator configuration in the WebSphere Business Monitor administrative console.

The Database Definition Language (DDL) scripts needed for creating the running database tables in the Historical database are stored in the **datamart.ddl** file on the root directory. To deploy the scripts, complete the following steps:

1. Open the DB2 **Command Window**. On UNIX, If the shell environment is configured, you can invoke the DB2 command line processor.
2. Change the path to the location of the script file.
3. Back up the Historical database before you deploy a new business measures model.
4. Run the command ***db2 terminate***. This ensures that any previous background processes which may be using a different code page value will not be used and that a new background process will be used to process this request.
5. Set the ***DB2CODEPAGE*** environment variable to 1208. The DB2 command line processor will by default interpret any character data using the current code page. The generated *datamart.ddl* however contains UTF-8 characters that will be corrupted unless the DB2CODEPAGE environment variable is set to 1208.
   a. On the UNIX operating system.
      - Using *sh*, *ksh*, *bash* type shells, run the command ***export DB2CODEPAGE=1208***.

- Using *csh*, *tsch* type shells, run the command ***setenv DB2CODEPAGE 1208***.

b. On the Windows operating system, run the command ***set DB2CODEPAGE=1208***.

6. Connect to the Historical database by running the command: ***db2 connect to <Historical_DB_Name>***. This script runs without automatically committing the changes.
7. Run the command ***db2 +c -stvf datamart.ddl > datamart.log***. This runs the script and saves a log file that records the transactions for troubleshooting purposes. Check the log file for any errors before you commit or rollback. If a rollback is required, run the command: ***db2 rollback*** to undo the actions. If no errors occurred, run the command ***db2 commit*** to commit the changes.
8. Disconnect from the Historical database after you run the script by running the command ***db2 disconnect <Historical_DB_Name>***.
9. Run the command ***db2 terminate*** to terminate the background processes.

Note: Under certain circumstances when running the ***datamart.ddl*** for a new version of an existing business measures model, you may see errors similar to the following: `SQL0605W The index was not created because the index "WBI.I_1133789461307" already exists with the required description. SQLSTATE=01550`. These errors can safely be ignored and you can commit the transaction, assuming no other errors occurred.

Deploying data movement services

Before the State to Runtime and Runtime to Historical data movement service can be deployed, the dynamic database tables should be created. Any error occurs while executing dynamic database tables creation scripts will result in problems during the data movement service deployment.

During schema generation, up to three compressed (ZIP files or JAR files) archives (named DS_State_setup, DS_Runtime_setup and DS_Datamart_setup) containing data movement service setup files will be created. The three archives will always be created the first time schema generation is performed for a business measures model. Subsequent generations, e.g. after a business measures model has been modified, can create zero, one, two or three new archives. An archive will only be created if a change in the existing replication environment is necessary to accommodate business measures model changes. The data movement service deployment archives are located in the directory that was specified in the Schema Generator Admin console configuration under the ″general″ tab.

Deployment of a data movement service involves creation and configuration of database objects in the source database (from where data is moved) as well as database objects in the target database (to where data is moved).

- *DS_State_setup* contains the deployment script for the source database setup of the State to Runtime data movement service.
- *DS_Runtime_setup* contains the deployment script for the target database setup of the State to Runtime data movement service and the deployment script for the source database setup of the Runtime to Historical data movement service.
- DS_Datamart_setup contains the deployment script for the target database setup of the Runtime to Historical data movement service.

1. Deploy the State to Runtime data movement service:

a. Determine on which machine you will be deploying the source artifacts for State to Runtime data movement service. In most cases this is the machine hosting the state database.
b. Create a working directory on that machine and copy (or transfer if the machine is remote) the generated DS_State_setup archive to that directory. You must choose a path with a length of 100 characters or less due to operating system dependent path length restrictions.
c. Extract the archive (in a .zip file on Windows and in a .jar file on UNIX) into the working directory.
d. During deployment, a variety of DB2 utilities will be used to create and configure database objects. To use those tools, the database environment must be set up. On Microsoft® Windows you can do that by opening a DB2 command window. On UNIX make sure that the appropriate environment variables are set.
e. Navigate to the directory into which you extracted the DS_State_setup archive.
f. Execute State_to_Runtime_setup_source.bat (the extension is .sh on UNIX) and follow the prompts. The script will display status messages indicating whether a certain command: succeeded, generated a warning or failed.
g. Inspect the generated log fileState_to_Runtime_setup_source.log for any warning or error messages. Do not proceed if any error messages are being displayed.
h. Backup the work directory. IBM support may use its content for troubleshooting.
i. Determine on which machine you will be deploying the target artifacts for State to Runtime data movement service. In most cases this is the machine hosting the Runtime database.
j. Create a working directory on that machine and copy (or transfer if the machine is remote) the generated DS_Runtime_setup archive to that directory. You must choose a path with a length of 100 characters or less due to operating system dependent path length restrictions.
k. Extract the archive (in a .zip file on Windows and in a .jar file on UNIX) into the working directory.
l. During deployment, a variety of DB2 utilities will be used to create and configure database objects. To use those tools, the database environment must be set up. On Microsoft Windows you can do that by opening a DB2 command window. On UNIX make sure that the appropriate environment variables are set.
m. Navigate to the directory into which you extracted the DS_Runtime_setuparchive.
n. Execute State_to_Runtime_setup_target.bat (the extension is .sh on UNIX) and follow the prompts. The script will display status messages indicating whether a certain command succeeded, generated a warning or failed.
o. Inspect the generated log file State_to_Runtime_setup_source.log for any warning or error messages. Do not proceed if any error messages are being displayed.
p. Backup the work directory. IBM support may use its content for troubleshooting.
q. If no problems were reported, the State to Runtime data movement service has been set up.

2. Deploy the Runtime to Historical data movement service:

a. Determine on which machine you will be deploying the source artifacts for the Runtime to Historical data movement service. In most cases this is the machine hosting the Runtime database. If you've deployed the target artifacts for the State to Runtime data movement service on the same machine, you can proceed with step **e** below because the necessary deployment files have already been extracted.
b. If deployment is performed on a machine other than the machine hosting the Runtime database, create a working directory on that machine and copy (or transfer if the machine is remote) the generated DS_Runtime_setup archive to that directory. You must choose a path with a length of 100 characters or less due to operating system dependent path length restrictions.
c. Extract the archive (in a .zip file on Windows and in a .jar file on UNIX) into the working directory.
d. During deployment, a variety of DB2 utilities will be used to create and configure database objects. To use those tools, the database environment must be set up. On Microsoft Windows you can do that by opening a DB2 command window. On UNIX make sure that the appropriate environment variables are set.
e. Navigate to the directory into which you extracted the DS_Runtime_setup archive.
f. Execute Runtime_to_Historical_setup_source.bat (the extension is .sh on UNIX) and follow the prompts. The script will display status messages indicating whether a certain command succeeded, generated a warning or failed.
g. Inspect the generated log file Runtime_to_Historical_setup_source.log for any warning or error messages. Do not proceed if any error messages are being displayed.
h. Backup the work directory. IBM support may use its content for trouble shooting.
i. Determine on which machine you will be deploying the target artifacts for the Runtime to Historical data movement service. In most cases this is the machine hosting the Historical database.
j. Create a working directory on that machine and copy (or transfer if the machine is remote) the generated DS_Datamart_setup archive to that directory. You must choose a path with a length of 100 characters or less due to operating system dependent path length restrictions.
k. Extract the archive (in a .zip file on Windows and in a .jar file on UNIX) into the working directory.
l. During deployment, a variety of DB2 utilities will be used to create and configure database objects. To use those tools, the database environment must be set up. On Microsoft Windows you can do that by opening a DB2 command window. On UNIX make sure that the appropriate environment variables are set.
m. Navigate to the directory into which you extracted the DS_Runtime_setup archive.
n. ExecuteRuntime_to_Historical_setup_target.bat (the extension is .sh on UNIX) and follow the prompts. The script will display status messages indicating whether a certain command succeeded, generated a warning or failed.

o. Inspect the generated log file State_to_Runtime_setup_source.log for any warning or error messages. Do not proceed if any error messages are being displayed.
p. Backup the work directory. IBM support may use its content for troubleshooting.
q. If no problems were reported, the Runtime to Historical data movement service has been set up for this business measures model.

Configuring data movement services options

For every Capture server that is created and configured by the data movement services component, there are two parameters that can affect the behavior of the Capture components. They are the `lag_time` and `startmode` parameters.

The `lag_time` and `startmode` parameters have the default values: ″7 days″ and ″WARMSI″. For more information on these parameters, refer to DB2 SQL Replication Guide and Reference.

You cannot modify these settings until the artifacts are deployed. However, you can alter these settings before you run any of the Capture component servers, or you can change the parameters for currently running Capture servers

Note: To enable changes performed while the Capture servers are running, you need to stop and restart the Capture servers.

If the default settings of the `lag_time` and `startmode` parameters are used while you are deploying replication artifacts, and a Capture server has been stopped for longer than 7 days and then restarted, the Capture component returns an error. The error states that the Capture server cannot run because the data is too old. You can overwrite the default in several ways. The following describes three methods:

1. Modify the default parameters as specified in the <CAPTURESERVERSCHEMA>.IBMSNAP_CAPPARMS table. After deploying the replication artifacts, you can determine the number of Capture servers that the data movement services component has created by running the following query against the Runtime database.

   ```
   CONNECT TO RUNTIME DATABASE
   SELECT DISTINCT OM_NAME, SERVICE_NAME, SRC_RM_CAP_SVR_NAME
   FROM WBIRMADM.RMMETADATA
   ORDER BY 1,2,3
   ```

   You see a table like the following:

Table 1. RMMETADATA example

OM_NAME	SERVICE_NAME	SRC_RM_CAP_SVR_NAME
SubDoctor3	Runtime to Historical	CAPTURE_18
SubDoctor3	State to Runtime	CAPTURE_1
SubDoctor3	State to Runtime	CAPTURE_115
SubDoctor3	State to Runtime	CAPTURE_156
SubDoctor3	State to Runtime	CAPTURE_194
SubDoctor3	State to Runtime	CAPTURE_212
SubDoctor3	State to Runtime	CAPTURE_250
SubDoctor3	State to Runtime	CAPTURE_41
SubDoctor3	State to Runtime	CAPTURE_59

Table 1. RMMETADATA example (continued)

OM_NAME	SERVICE_NAME	SRC_RM_CAP_SVR_NAME
SubDoctor3	State to Runtime	CAPTURE_97

The OM_NAME is the name of the WebSphere Business Modeler project. The SERVICE_NAME indicates the data movement service, and the SRC_RM_CAP_SVR_NAME is the identifier (CAPTURE SCHEMA) for the Capture server that is being used as part of the data movement service. In the above table, there is one capture server for the Runtime to Historical data movement service and nine for the State to Runtime data movement service.

Note: The number of servers and the names of the servers vary, based on the model being used and the policy parameters specified during artifact generation.

The database services component supports all the options for `lag_time` and `startmode` parameters, but you should be aware that there are serious performance problems if the number of cold starts (the start of the Capture servers after failure) increases. If there are frequent cold starts, the ETL component of the data movement service processes all existing records instead of simply recording the changes. Once all of the Capture servers that need to be modified are identified, you can modify the default parameters in the database. Once the default parameters are modified for each Capture server, you can start the Capture servers.

2. Or modify the command line that starts the Capture server. During generation of the database services artifact, convenience scripts are generated that can start and stop the Capture and Apply servers. The capture start scripts (StartCapture_#.bat or StartCapture_#.sh) are located in the directory <data_movement_service_name>\source. Each of these scripts contains the **asncap** command, which is used to start the Capture program. For more information about these parameters, refer to IBM DB2 documentation. Modify the start script accordingly, save, and then run the start script to run the Capture server with the new settings.
3. Or modify the Capture server during runtime. Follow the instructions found in the IBM DB2 documentation, which describes how to temporarily change the settings for a running Capture server.

Finalizing data movement services setup

The deployed Capture and Apply component instances are using by default the credentials of the user who started them. While this may be sufficient for some topologies, there are two scenarios where alternate credentials have to be used

- **First Scenario - Alternate user credentials:** The database administrator (DBA) wants to be logged on as user *user1* but wants the utility to use user *user2* to move data from the source database to the target database.
- **Second Scenario - Distributed environment:** The DBA plans to run the utility on *machine1*. The source or the target database is maintained on another machine *machine2*.

To support those scenarios, you must create password files containing the user credentials to be used instead of the current user credentials. Because password files are not automatically created during deployment, you need to perform the following steps for the two scenarios:

1. Prepare a file to store the *user id* and *password* information to be used when connecting to a source database. In a DB2 command-line window, enter the

following command, and substitute any placeholders marked like this: <*place_holder_name*> with the appropriate value.

`asnpwd INIT encrypt all using <password_file>.` The asnpwd tool creates an empty file: <*password_file*>.

Example invocations: `asnpwd INIT encrypt all using password.aut`

2. Save the database access information (user ID, password, and database name) for each database the replication utility will have to connect to. In a DB2 command-line window, enter the following command, and substitute any placeholders marked like this: <*place_holder_name*> with the appropriate value.

   `asnpwd ADD alias <DB_name> ID <user_ID> PASSWORD <Password> using <password_file>.`

   Repeat this step for each database if necessary. The program encrypts the information you enter and saves it in the <*password_file*>.

   Example invocation:

   - `asnpwd ADD alias STMD7 id MYUSRID password MYPASSWRD using password.aut`
   - `asnpwd ADD alias RTMD7 id MYUSRID2 password MYPASSWRD2 using password.aut`

3. Update the utilities configuration files by modifying the generated executable startup scripts (StartCapture and StartApply). You append the password-file parameter to the command-line invocation of the replication utility. The utility uses the encrypted user credentials that are stored in the specified file instead of the default credentials. The password file must be located in the work directory defined by the CAPTURE_PATH (or APPLY_PATH) parameter.

   Example of changes:

   - Original file content of Capture start script: `db2cmd asncap CAPTURE_SERVER=stmd7 CAPTURE_SCHEMA=CAPTURE_1 CAPTURE_PATH="c:\tmp\state_capture_log"`
   - Modified file content of Capture start script: `db2cmd asncap CAPTURE_SERVER=stmd7 CAPTURE_SCHEMA=CAPTURE_1 CAPTURE_PATH="c:\tmp\state_capture_log" pwdfile="password.aut"`
   - Original file content of Apply start script: `db2cmd asnapply APPLY_QUAL=Apply_1 CONTROL_SERVER=RTMD7 APPLY_PATH="C:\tmp\apply"`
   - Modified file content of Apply start script: `db2cmd asnapply APPLY_QUAL=Apply_1 CONTROL_SERVER=RTMD7 APPLY_PATH="C:\tmp\apply" pwdfile="password.aut"`

4. Copy the<*password_file*> that was created in steps 1 and 2 into the appropriate directory. The replication utilities attempt to open the password file upon startup. An error occurs if the file: <*password_file*> does not exist in the work directory identified by the CAPTURE_PATH (or APPLY_PATH) parameter. If no work-directory parameter is specified, the utilities attempt to locate the file in the current working directory.

For more information on DB2 utilities, refer to DB2 SQL Replication Guide and Reference.

Consolidating start and stop scripts

To simplify the process of starting and stopping a data movement service, you can consolidate the generated start and stop scripts and invoke them through master scripts.

Because the Capture and Apply components have to be run on the system where the databases are located, consolidation options vary depending on the topology used. Regardless of how the scripts are consolidated, to prevent initialization errors, you need to ensure that no two component instances are launched simultaneously.

Although you can start or stop each Capture or Apply Component instance separately, it is convenient to consolidate the content of all component instance start and stop scripts so that only one script is needed to start or stop the data movement service for a single business measures model. You can consolidate scripts by:

1. Identifying the Capture component instances start and stop scripts for the source database.
2. Creating Capture master start and stop scripts that invoke the Capture component instances start and stop scripts for the source database.
3. Identifying the Apply component instances start and stop scripts for the target database.
4. Creating Capture master start and stop scripts that invoke the Capture component instances start and stop scripts for the target database.

As a result of this consolidation, only four start (or stop) scripts need to be executed to start or stop the data movement services for a business measures model.

You can consolidate further if there is no need to start or stop the two data movement services separately. In this case, only three start and stop scripts are necessary:

- A script starts (stops) all Capture Component instances in the State database.
- A script starts (stops) all Capture Component instances and Apply Components in the Runtime database.
- A script starts (stops) all Apply Components in the Historical database.

If all three databases reside on one system, these three consolidated scripts can be further consolidated to just a single script that starts or stops all Capture and Apply Component instances.

There is a case where consolidation is required of the start and stop scripts for a data movement service that has been created by different deployments. During the initial deployment of a data movement service, start and stop scripts for all business measures groups are created. Subsequent deployments that are a result of changes to the business measures model do not contain start and stop scripts for existing business measures groups. Instead only start and stop scripts for new business measures groups will be made available. You need to manually update previously created consolidated start and stop scripts.

The following example illustrates this case: An initial data-movement-service deployment for a business measures model *FinanceModel* contains three business measures groups. Three Capture start and stop scripts have been created for the State database. Subsequently, the model is updated and a new business measure added. Only one Capture start and stop script for the new business measures group will be created during deployment. Four Capture start and stop scripts need to be run to enable the data movement service.

Starting and stopping a data movement service

A data movement service for a given business measures model is started and stopped by starting and stopping the associated Capture and Apply component instances. During the data movement service deployment, the start and stop scripts have been created, so that, those scripts can be used to start and stop the data movement services.

Your topology determines on which machines the component instances should be running. In general, Capture components instances must run on the machine that hosts the State database (for the State to Runtime Data Movement Service) and on the machine hosting the Runtime database (for the Runtime to Historical Data Movement Service). The Apply component instances should run on the machine that hosts the Runtime database (for the State to Runtime Data Movement Service) and on the machine hosting the Historical database (for the Runtime to Historical Data Movement Service). In this configuration, Apply component instances will pull data from the source database which results in better performance than if they would reside on the machine hosting the State database (for the State to Runtime Data Movement Service) and on the machine hosting the Runtime database (for the Runtime to Historical Data Movement Service).

The following information explains how to start the State to Runtime data movement service and the Runtime to Historical data movement service. It also describes how to stop the State to Runtime data movement service and the Runtime to Historical data movement service.

Note: The State to Runtime and Runtime to Historical data movement services are independent of each other. However, it is preferable to start the State to Runtime service prior to starting the Runtime to Historical Service. In some instances, it may be preferable to start the Runtime to Historical data movement service after the Monitor Server has processed entries for business measures model and after the State to Runtime data movement service has populated the Runtime database tables supporting this model. This can get information into the Historical database faster than waiting for the Runtime to Historical data movement services interval.

Starting the State to Runtime data movement service

The deployment archives DS_State_setup and DS_Runtime_setup contains executable scripts that can be used to start the Capture and Apply component instances for the State to Runtime data movement service. If the archive was created as a result of a change to the business measures model, only start scripts for new Capture and Apply component instances have been packaged.

Note: You can consolidate the scripts to starting the data movement service. For more information about consolidating scripts refer to “Consolidating start and stop scripts” on page 49.

However, the instructions below can be used even if no consolidation has been performed. To start the State to Runtime data movement service for a given business measures model:

1. Identify all Capture component instances that have been assigned to the business measures model in the State database.

   If you have already consolidated all Capture component instance start scripts, nothing needs to be done. Proceed with the next step. If you have not consolidated the scripts yet (and do not wish to consolidate them) you need to identify all capture component instances that have ever been created for this

business measures model. The Capture component instance start scripts are automatically generated the first time schema generation is performed for a business measures model. Subsequent schema generations (e.g. after you have updated the business measures model) only generate start scripts for new Capture component instances. To identify all relevant start scripts you will have to repeat the following steps for each deployment that you have performed for this business measures model.

a. Navigate to the directory where you performed a deployment for this model.
b. Navigate to the State_to_Runtime\source subdirectory and locate all StartCapture_<number>scripts.
c. Repeat above steps for each deployment of this business measures model.

2. Start Capture component instances

   The identified Capture component instances must be started on the machine hosting the State database. If the start scripts have been consolidated, launch the consolidated start script. If no consolidation has been performed, you need to execute each one of the start scripts that you identified in the previous step. The scripts should not be launched simultaneously, because the Capture utility initialization could otherwise fail. However, it does not matter in which order the start scripts are being launched. Authorization requirements, the user id that starts the Capture component instances should hold:

   - Database Administration Authority (DBADM) authority on the State database.
   - Write access to the directory referenced by the *CAPTURE_PATH* parameter in the start scripts.
   - Read access to the file referenced by the optional parameter *PWDFILE* in the start scripts

3. Identify all Apply component instances that have been assigned to the business measures model in the Runtime database.

   If you have already consolidated all Apply component instance start scripts, nothing needs to be done. Proceed with the next step. If you have not consolidated the scripts yet (and do not wish to consolidate them) you need to identify all Apply component instances that have ever been created for this business measures model. Apply component instance start scripts are automatically generated the first time schema generation is performed for a business measures model. Subsequent schema generations (e.g. after you have updated the business measures model) only generate start scripts for new Apply component instances. To identify all relevant start scripts you will have to repeat the following steps for each deployment that you have performed for this business measures model:

   a. Navigate to the directory where you performed a deployment for this model
   b. Navigate to the State_to_Runtime\targetsubdirectory and locate all StartApply_<number> scripts.
   c. Repeat above steps for each deployment of this business measures model.

4. Start Apply Component instances

   The identified Apply component instances should be started on the machine hosting the Runtime database. If the start scripts have been consolidated, launch the consolidated start script. If no consolidation has been performed, you need to execute each one of the start scripts that you identified in the previous step. The scripts should not be launched simultaneously, because the Apply utility initialization could otherwise fail. However, it does not matter in

which order the start scripts are being launched. Authorization requirements, the user id that starts the Apply component instances must have:

- SELECT/INSERT/UPDATE/DELETE privileges for the associated Capture component instance control tables in the State database
- SELECT privileges for the associated Capture component instance work tables in the State database.
- SELECT/INSERT/UPDATE/DELETE privileges for the associated replication staging tables in the Runtime database.
- SELECT/INSERT/UPDATE/DELETE privileges for the Apply component instance control tables in the Runtime database.
- Write access to the directory referenced by the *APPLY_PATH* parameter in the start scripts.
- Read access to the file referenced by the optional parameter *PWDFILE* in the start scripts

5. Verify that each one of the Capture and Apply component instances started successfully.

Starting the Runtime to Historical data movement service

The deployment archives DS_Runtime_setup and DS_Datamart_setup contains executable scripts that can be used to start the Capture and Apply component instances for the Runtime to Historical Data Movement Service. If the archive was created as a result of a change to the business measures model, only start scripts for new Capture and Apply component instances have been packaged.

Note: You can consolidate the scripts to starting the data movement service. For more information about consolidating scripts refer to "Consolidating start and stop scripts" on page 49.

However, the instructions below can be used even if no consolidation has been performed. To start the Runtime to Historical data movement service for a given business measures model:

1. Identify all Capture component instances that have been assigned to the business measures model in the Runtime database.

   If you have already consolidated all Capture component instance start scripts, nothing needs to be done. Proceed with the next step. If you have not consolidated the scripts yet (and do not wish to consolidate them) you need to identify all capture component instances that have ever been created for this business measures model. The Capture component instance start scripts are automatically generated the first time schema generation is performed for a business measures model. Subsequent schema generations (e.g. after you have updated the business measures model) only generate start scripts for new Capture component instances. To identify all relevant start scripts you will have to repeat the following steps for each deployment that you have performed for this business measures model.

   a. Navigate to the directory where you performed a deployment for this model.
   b. Navigate to the Runtime_to_Historical\source subdirectory and locate all StartCapture_<number>scripts.
   c. Repeat above steps for each deployment of this business measures model.

2. Start Capture component instances.

   The identified Capture component instances must be started on the machine hosting the Runtime database. If the start scripts have been consolidated, launch the consolidated start script. If no consolidation has been performed,

you need to execute each one of the start scripts that you identified in the previous step. The scripts should not be launched simultaneously, because the Capture utility initialization could otherwise fail. However, it does not matter in which order the start scripts are being launched. Authorization requirements, the user id that starts the Capture component instances should hold:

- Database Administration Authority (DBADM) authority on the Runtime database.
- Write access to the directory referenced by the *CAPTURE_PATH* parameter in the start scripts.
- Read access to the file referenced by the optional parameter *PWDFILE* in the start scripts

3. Identify all Apply component instances that have been assigned to the business measures model in the Historical database.

   If you have already consolidated all Apply component instance start scripts, nothing needs to be done. Proceed with the next step. If you have not consolidated the scripts yet (and do not wish to consolidate them) you need to identify all Apply component instances that have ever been created for this business measures model. Apply component instance start scripts are automatically generated the first time schema generation is performed for a business measures model. Subsequent schema generations (e.g. after you have updated the business measures model) only generate start scripts for new Apply component instances. To identify all relevant start scripts you will have to repeat the following steps for each deployment that you have performed for this business measures model:

   a. Navigate to the directory where you performed a deployment for this model
   b. Navigate to the Runtime_to_Historical\targetsubdirectory and locate all StartApply_<number> scripts.
   c. Repeat above steps for each deployment of this business measures model.

4. Start Apply instances.

   The identified Apply component instances should be started on the machine hosting the Historical database. If the start scripts have been consolidated, launch the consolidated start script. If no consolidation has been performed, you need to execute each one of the start scripts that you identified in the previous step. The scripts should not be launched simultaneously, because the Apply utility initialization could otherwise fail. However, it does not matter in which order the start scripts are being launched. Authorization requirements, the user id that starts the Apply component instances must have:

   - SELECT/INSERT/UPDATE/DELETE privileges for the associated Capture component instance control tables in the Runtime database
   - SELECT privileges for the associated Capture component instance work tables in the Runtime database.
   - SELECT/INSERT/UPDATE/DELETE privileges for the associated replication staging tables in the Historical database.
   - SELECT/INSERT/UPDATE/DELETE privileges for the Apply component instance control tables in the Historical database.
   - Write access to the directory referenced by the *APPLY_PATH* parameter in the start scripts.
   - Read access to the file referenced by the optional parameter *PWDFILE* in the start scripts

5. Verify that each one of the Capture and Apply component instances started successfully.

Stopping the State to Runtime data movement service

The process of stopping the Runtime to Historical Data Movement Service is very similar to the process of starting it. The deployment archives DS_State_setup and DS_Runtime_setup contain executable scripts that can be used to stop the Capture and Apply component instances for the Runtime to Historical Data Movement Service.

If the archive was created as a result of a change to the business measures model, only stop scripts for new Capture and Apply component instances have been packaged.

Note: It is recommended to consolidate the scripts prior to stopping the data movement service. For more information about replication scripts consolidation, refer to "Consolidating start and stop scripts" on page 49.

However, the instructions below can be used even if no consolidation has been performed.

To stop the State to Runtime Data Movement Service for a given business measures model:

1. Identify all Capture component instances that have been assigned to the business measures model in the State database. If you have already consolidated all Capture component instance stop scripts, nothing needs to be done. Proceed with the next step. If you have not consolidated the scripts yet (and do not wish to consolidate them) you need to identify all capture component instances that have ever been created for this business measures model. The Capture component instance stop scripts are automatically generated the first time schema generation is performed for a business measures model. Subsequent schema generations (e.g. after you have updated the business measures model) only generate stop scripts for new Capture component instances. To identify all relevant stop scripts you will have to repeat the following steps for each deployment that you have performed for this business measures model.
   a. Navigate to the directory where you performed a deployment for this model
   b. Navigate to the State_to_Runtime\source subdirectory and locate all StopCapture_<number> scripts.
   c. Repeat above steps for each deployment of this business measures model.
2. Stop Capture component instances. The identified Capture component instances must be stopped on the machine hosting the State database. If the stop scripts have been consolidated, launch the consolidated stop script. If no consolidation has been performed, you need to execute each one of the stop scripts that you identified in the previous step. The order in which the stop scripts are being launched does not matter.

   Note: The stop scripts work asynchronously and occasionally, there may be a delay between when a stop command is issued and when the capture component stops. This is due to the fact that the Capture component instance is finishing up a transaction prior to stopping.
3. Identify all Apply component instances that have been assigned to the business measures model in the Runtime database. If you have already consolidated all Apply component instance stop scripts, nothing needs to be done. Proceed with the next step. If you have not consolidated the scripts yet (and do not wish to consolidate them) you need to identify all Apply component instances that

have ever been created for this business measures model. The Apply component instance stop scripts are automatically generated the first time schema generation is performed for a business measures model. Subsequent schema generations (e.g. after you have updated the business measures model) only generate stop scripts for new Apply component instances. To identify all relevant stop scripts you will have to repeat the following steps for each deployment that you have performed for this business measures model:

a. Navigate to the directory where you performed a deployment for this model.
b. navigate to the State_to_Runtime\target subdirectory and locate all StopApply_<number>scripts.
c. Repeat above steps for each deployment of this business measures model.

4. Stop Apply component instances.

   The identified Apply component instances should be stopped on the machine hosting the Runtime database. If the stop scripts have been consolidated, launch the consolidated stop script. If no consolidation has been performed, you need to execute each one of the stop scripts that you identified in the previous step. The stop scripts can be launched in any order.

   Note: The stop scripts work asynchronously and occasionally, there may be a delay between when a stop command is issued and when the apply component stops. This is due to the fact that the Capture component instance is finishing up one or more transactions prior to stopping.

Stopping the Runtime to Historical data movement service

The process of stopping the Runtime to Historical Data Movement Service is very similar to the process of starting it. The deployment archives DS_Runtime_setup and DS_Datamart_setup contain executable scripts that can be used to stop the Capture and Apply component instances for the Runtime to Historical Data Movement Service.

If the archive was created as a result of a change to the business measures model, only stop scripts for new Capture and Apply component instances have been packaged.

Note: It is recommended to consolidate the scripts prior to stopping the data movement service. For more information about replication scripts consolidation, refer to "Consolidating start and stop scripts" on page 49.

However, the instructions below can be used even if no consolidation has been performed.

To stop the Runtime to Historical Data Movement Service for a given business measures model:

1. Identify all Capture component instances that have been assigned to the business measures model in the Runtime database. If you have already consolidated all Capture component instance stop scripts, nothing needs to be done. Proceed with the next step. If you have not consolidated the scripts yet (and do not wish to consolidate them) you need to identify all capture component instances that have ever been created for this business measures model. The Capture component instance stop scripts are automatically generated the first time schema generation is performed for a business measures model. Subsequent schema generations (e.g. after you have updated the business measures model) only generate stop scripts for new Capture

component instances. To identify all relevant stop scripts you will have to repeat the following steps for each deployment that you have performed for this business measures model.

a. Navigate to the directory where you performed a deployment for this model
b. Navigate to the Runtime_to_Historical\source subdirectory and locate all StopCapture_<number> scripts.
c. Repeat above steps for each deployment of this business measures model.

2. Stop Capture component instances. The identified Capture component instances must be stopped on the machine hosting the Runtime database. If the stop scripts have been consolidated, launch the consolidated stop script. If no consolidation has been performed, you need to execute each one of the stop scripts that you identified in the previous step. The order in which the stop scripts are being launched does not matter.

   Note: The stop scripts work asynchronously and occasionally, there may be a delay between when a stop command is issued and when the capture component stops. This is due to the fact that the Capture component instance is finishing up a transaction prior to stopping.

3. Identify all Apply component instances that have been assigned to the business measures model in the Historical database. If you have already consolidated all Apply component instance stop scripts, nothing needs to be done. Proceed with the next step. If you have not consolidated the scripts yet (and do not wish to consolidate them) you need to identify all Apply component instances that have ever been created for this business measures model. The Apply component instance stop scripts are automatically generated the first time schema generation is performed for a business measures model. Subsequent schema generations (e.g. after you have updated the business measures model) only generate stop scripts for new Apply component instances. To identify all relevant stop scripts you will have to repeat the following steps for each deployment that you have performed for this business measures model:

   a. Navigate to the directory where you performed a deployment for this model.
   b. navigate to the Runtime_to_Historical\target subdirectory and locate all StopApply_<number>scripts.
   c. Repeat above steps for each deployment of this business measures model.

4. Stop Apply component instances.

   The identified Apply component instances should be stopped on the machine hosting the Runtime database. If the stop scripts have been consolidated, launch the consolidated stop script. If no consolidation has been performed, you need to execute each one of the stop scripts that you identified in the previous step. The stop scripts can be launched in any order.

   Note: The stop scripts work asynchronously and occasionally, there may be a delay between when a stop command is issued and when the apply component stops. This is due to the fact that the Capture component instance is finishing up one or more transactions prior to stopping.

Deploying Cube Views database schema

The Schema Generator produces Cube Views metadata in an XML file format. It represents the DB2 Cube Views definitions that correspond to the business measures model. The cube views definitions are deployed on both Windows and AIX platforms.

Deploying Cube Views database schema on the Windows platform

The Cube Views metadata is stored in the Schema Generator output folder. This output folder is defined by the user through the WebSphere Business Monitor administrative console.

To deploy the Cube Views metadata file, complete the following steps:

1. Start the DB2 OLAP Center. The **DB2 database connection** dialog box appears.
2. In the **DB2 database connection** dialog box, do the following:
   a. In the **Database name** field, type the name of the Historical database.
   b. In the **User name** field, type the user ID of a user who has administrative rights to the database.
   c. In the **Password** field, type the password of a user who has administrative rights to the database.
   d. Click **OK**.
   e. The first time you connect to the database, a message may inform you that the database needs to be configured for Cube Views. Click **Yes** on the message to start the initialization and configuration.
3. In the OLAP Center window, import the Cube Views metadata file as follows:
   a. From the menu, select **OLAP Center → Import**. The Import wizard starts.
   b. Select the Cube Views XML file, which is stored in the Schema Generator output folder. The name of the file is *model_cv.xml*.
   c. Click **Finish**. The import process starts.
4. After import is completed, on the **Import Options** page of the **Import Wizard** window, click **Finish**.

Deploying Cube Views database schema on the AIX platform

The Cube Views metadata is stored in the Schema Generator output folder. This output folder is defined by the user through the WebSphere Business Monitor administrative console.

To deploy the Cube Views metadata file, complete the following steps:

1. Open the DB2 **Command Window** editor.
2. Connect to the Historical database with the database instance user (example: db2inst1) by running the command: **db2 connect to HISTORICAL_database_name**.
3. Change directory to **<DB2_INST_HOME>/sqllib/misc** directory, and then run the command: **db2 -tvf db2mdapi.sql**.
4. Run the command: **db2mdapiclient -d HISTORY -i <GENERATION_DIR>/schemagen/import_model.xml -m <GENERATION_DIR>/schemagen/model_cv.xml> -u <userid> -p <pw> -o <GENERATION_DIR>/schemagen/myoutput.xml**.

   Where

   - -d is the Historical database name.
   - -i is the Schema Generator produced import_model.xml file.
   - -u is the user ID.
   - -p is the password.
   - -o is the output operation file name where DB2 output information is stored.

- -m is the input metadata command or instruction to DB2. The Schema-Generator produces model_cv.xml file which is be used as the multidimensional metadata.
- <GENERATION_DIR> refers to the output directory where the Schema Generator stores the artifacts that it generates.

Example:

```
su - db2inst1
db2 connect to HISTORY
cd /home/db2inst1/sqllib/misc
db2 -tvf
db2mdapi.sql
db2mdapiclient -d HISTORY
-i /opt/IBM/WebSphere/Monitor/generation/schemagen/import_model.xml
-m /opt/IBM/WebSphere/Monitor/generation/schemagen/model_cv.xml
-u db2inst1 -p monPa55w -o /tmp/import_output.xml
```

Creating ABX cubes manually

You create the ABX cubes manually on the machine that has the IBM DB2 ALPHABLOX server installed on it. These cubes will be used by the WebSphere Business Monitor dashboards.

After you deploy the cube-views definitions, but before you use the dashboards, complete the following steps:

1. Point your web browser to: http://<hostname>:9081/AlphaBlox/home/Admin, and log in to IBM DB2 ALPHABLOX admin console.
2. Select the **ADMINISTRATION** tab.
3. Click **Cubes**.
4. To create a cube, click **Create**.
   a. From the **Relational Data Source** list, select the appropriate Historical database defined at installation.
   b. Check the **Enabled** check box beside the **DB2 AlphaBlox Cube Name**.
   c. Check the **Enable DB2 Cube Views Settings** check box. Wait a few seconds until the fields become visible.
5. For each cube that is defined under the **Cube Model**, you must create a cube.
   a. From the **Cube Model** list, select the cube model.
   b. From the **Cube** list, select the cube. There is only one cube per cube model.
   c. In the **DB2 AlphaBlox Cube Name** field, type the cube name. The name should be exactly as it appears in the **Cube** list. Example: CISS.NOOP. Do not include WBI, which is the schema name.
   d. Select the **Use Business Names** option.
   e. Click **Import Cube Definition**, and wait while processing takes place.
6. Click **OK** to save the cube.
7. Repeat step 5 (cube creation) for each cube that exists.

Populating dimensional tables manually

You may have existing data that will be used as dimensional data (for example, a database of customer information that should be populated into the customer dimension). You can use the Historical database to manually populate the dimensional tables with this data.

There are several things to be aware of as you populate the tables.

- Pay careful attention when you create the dimension in WebSphere Business Modeler so that you can populate the dimension with existing data. Make sure that your dimension as defined in WebSphere Business Modeler contains the appropriate metrics with appropriate data types so that you can store existing data into the dimensional table created by the Schema Generator.
- When inserting data manually, use negative values for the SK_<> column. This is the surrogate key for the table. Positive surrogate key values are used by data services when it is populating these tables; to avoid collisions, you must use negative values.
- When inserting data into the dimension table, make sure that no column is set to the NULL value. If there is no meaningful value to insert into a given column, you must select a meaningful default and use it. Never insert NULL into this table. The empty string ("") is acceptable for string data types, however.
- As you map new process instances to the dimension data you have inserted, a process instance may have no match to the existing data (for example, a process associated with a new customer who is not currently listed in the dimension table). In that case, a new row will be created in the table for this set of data. The table now contains data you entered and this other data.
- A non-key attribute of a dimension is updated as new data comes in. For example, suppose you have a customer dimension in which the key metric is "CustomerName", and a non-key metric is "CreditLimit". Initially, this table might contain the row ['Widgets, Inc',50000] from your existing customer data. If a new event is processed that contains a CreditLimit for 'Widgets, Inc' of 75,000, the Customer dimension table row will be updated to ['Widgets, Inc', 75000]. This update only occurs when the key metrics match an existing row and the non-key metrics do not. In these cases, the non-key values are updated to reflect the new data.

To establish which dimension table corresponds to the dimension you are manually populating, and which columns within the table correspond to the various dimension attributes, use the *datamartMapping.txt* text file, which is located in the Schema Generator output directory (after you run the Schema Generator).

Historical database schema

The database schemas describe the database tables and the relations among them. Using the database schemas, you can plan for the size of the database.

The information in the Historical database schemas helps you understand the mapping between the imported business measures model and the database tables. The dashboards use the Historical database for multidimensional analysis and generating reports.

Note:

- The Repository, State, and Runtime databases are for internal use only, and they are subject to change without notice.
- Customer written custom code that directly accesses the State, Runtime, or Repository databases is not supported by IBM.
- You cannot create your own dashboards using the Historical database schema.

The Historical database is initially populated with date/time data ranging from the year 1995 through the year 2009. If you anticipate recording dates/times (either as process start/termination times or as other metric data) which is outside this range of dates, you should use the following SQL script to add additional dates to the DIM_TIME table in the Historical database:

```
insert into <your WBI schema name>.dim_time( surrogate_key, year, month, day)
with WBITIME (skey, ldate) as
(select surrogate_key+1 as skey,
  COALESCE(
   DATE(SUBSTR(DIGITS(YEAR),7,4) || '-' ||
    SUBSTR(DIGITS(MONTH),4,2) || '-' ||
    SUBSTR(DIGITS(DAY), 4,2)) + 1 DAYS,
   DATE('YYYY-MM-DD of the first day you'd want to start from,
      in case the DIM_TIME table is empty.')
  )as ldate
 from sysibm.sysdummy1, <your WBI schema name>.dim_time
 where
    DATE(
     SUBSTR(DIGITS(YEAR) ,7,4) || '-' ||
     SUBSTR(DIGITS(MONTH),4,2) || '-' ||
     SUBSTR(DIGITS(DAY)  ,4,2)
       ) =
 (
  SELECT
    MAX(
    DATE(SUBSTR(DIGITS(YEAR),7,4) || '-' ||
    SUBSTR(DIGITS(MONTH),4,2) || '-' ||
    SUBSTR(DIGITS(DAY), 4,2)))
     FROM <your WBI schema name>.DIM_TIME
 )
 UNION ALL
 SELECT parent.skey+1, ldate + 1 DAYS
 from WBITIME parent
 where YEAR(ldate + 1 days) <   where YEAR(ldate + 1 days) <
  <YYYY 4 Digit YEAR FOR WHICH YOU DON't WANT DATA to end in>
)
select a.skey, year(a.ldate), month(a.ldate), day(a.ldate)
from WBITIME a
```

```
WHERE
a.ldate >= DATE('YYYY-MM-DD: The start of the range that should be inserted.')
AND a.ldate <= DATE('YYYY-MM-DD: The end of the range that should be inserted.')
```

Note: There are four locations in this script which be updated to specify the beginning and ending dates for the data you wish to insert into DIM_TIME. There are also three locations in which you must specify your WBI Schema Name (typically ″WBI″)

Database services

This reference information will help you to work with the database services.

Historical database schema

The Historical database tables are divided into two types. They are the static tables that are created at WebSphere Business Monitor installation time, and the dynamic tables that are created for each imported business measures model.

The description of both types of the Historical database tables and the corresponding mapping of each column to the business measures model is listed in the following tables.

Note:

- Nullable: means this column can either accept or not accept null values
- Descriptor: describes the mapping between a column and the business measures model definitions. Each column does not have to have a descriptor.

Static database tables

DIM_TIME

The time dimension table.

Column name	Column Type	Column Description	Nullable
SURROGATE_KEY	INTEGER	This is the primary key	N
DAY	SMALLINT	Represents the day	N
MONTH	SMALLINT	Represents the month	N
YEAR	INTEGER	Represents the year	N

Dynamic database tables

The Historical database implements a star schema structure with a central Fact table surrounded by multiple Dimension ″leaf″ tables. The fact table is similar to the context table in the State and Runtime databases. There is one star for the context, and one for the context's corresponding activities. For example, there can be one context instance table, and one activity instance table per context in the State and Runtime databases.

The Context Fact table

Naming convention: FCT_<machine generated name of context>

The columns that always exist are:

Column name	Column Type	Column Description	Nullable
MCI_MCIID	DECIMAL(19,0)	Unique identifier of the activity instance and also the primary key of the table.	N

The Context Fact table

Naming convention: FCT_<machine generated name of context>

The columns that always exist are:

Column name	Column Type	Column Description	Nullable
PARENT_MCIID	DECIMAL(19,0)	Unique identifier of the parent process instance, if any.	Y
SK_<machine generated name of the dimension>	INTEGER	Foreign key pointing to a dimension table. FK relationship is defined. One of these columns is defined for each dimension present in the context.	Y
GMT_<machine generated name of metric>	TIMESTAMP	A timestamp value used to store the GMT time value of any timestamp metric data types. (This column is created only when the time metric is marked as a Dimension. When Time metrics are marked as dimensions, they are only stored with day, month, year granularity, so this column provides the ability to see the exact time value of these metrics.)	Y

The following three column types are used when metrics are marked as Facts (not dimensions).

Column types that are used when metrics are marked as Facts

Column name	Column Type	Column Description	Nullable
M_<machine generated name>	Data type varies with the data type defined in the business measures model.	Used to represent a metric or Keydefinition value.	Y
C_<machine generated name>	BIGINT	Used to represent counters.	Y
T1_<machine generated name>	BIGINT	Used to represent accumulated time for timers. (Timers are represented with a single column in the Historical database; they use multiple columns in the State database.)	Y

The Dimension table

Zero or more dimension tables will be defined for each context, depending how many dimensions the context defines. Typically there will be at least a Time dimension.

Naming Convention: DIM_<machine generated name of dimension>

The columns which are always defined:

Column name	Column Type	Column Description	Nullable
SURROGATE_KEY	INTEGER	Machine generated primary key value for this dimension row. PK is defined.	N

These are the columns that are definition based. The dimension table contains a column for each metric that is defined as part of this dimension.

The columns that are definition based

Column name	Column Type	Column Description	Nullable
M_<machine generated name>	Data type varies with the data type defined in the business measures model.	Used to represent a metric or key definition value.	Y
C_<machine generated name>	BIGINT	Used to represent counters.	Y
T1_<machine generated name>	BIGINT	Used to represent accumulated time for timers. (Timers are represented with a single column in the Historical database, whereas they use multiple columns in the State database).	Y

The activity star schema follows the same conventions, except that the tables are named AFC_ and ADM_ respectively.

Data movement service control table

This section describes the Data Movement Services control table structure. Each one of the State, Runtime, and Historical databases contain two control tables that can be manipulated to configure the behavior of the local data-control movement service components. The control tables are static tables.

RMCONTROL

Contains configuration settings specific to the behavior of ETL component instances. This table is only populated and used in the Runtime and Historical databases, because no ETL component is required in the State database. Each row in this table corresponds to one target table that needs to be populated. Altering column values for a given row will only affect the ETL component instance that has been assigned to populate that target table.

Column name	Column Type	Column Description	Nullable
TARGETTABLE	CHARACTER	The fully qualified table name of the target table that is to be populated by the stored procedure controlled by this entry.	N
COMMITINTERVAL	NUMERIC	The commit interval used by the stored procedure when a cursor is used to insert rows into the target table.	Y
LOGLEVEL	NUMERIC	The logging level that determines how much information a stored procedure will put in the WBIRMADM.RMLOG table. Valid values are 0 and 1. 0 equals minimal logging, and 1 equals maximum logging.	Y
LASTSEQUENCE	CHARACTER	The last SEQUENCE value processed by the ETL stored procedure from the staging table. This column is updated by the stored procedure at runtime.	N
LASTUPDATED	TIMESTAMP	The last time a scheduled invocation was made. This column is controlled by the stored procedure and is only used for scheduling purposes.	Y
NEXTSTARTTIME	TIMESTAMP	The next time after which an ETL invocation will be made.	Y
ETLSCHEDMETHOD .	NUMERIC	The scheduling method to be used. Only 0 is a valid value.	Y
ETL_0_MINUTES	NUMERIC	The number of minutes that should elapse between scheduled ETL runs.	Y
TGT_RM_SPETL_NAME	CHARACTER	The fully qualified stored procedure name that is responsible for populating the TARGETTABLE.	Y

Do not alter any of the following column values or unexpected behavior will occur:

- TARGETTABLE
- LASTSEQUENCE
- LASTUPDATED
- ETLSCHEDMETHOD
- TGT_RM_SPETL_NAME

Changes to the following columns will be committed the next time an ETL component instance is invoked:

- COMMITINTERVAL
- NEXTSTARTTIME
- LOGLEVEL
- ETL_0_MINUTES

RMPRUNECTRL

Contains configuration settings specific to the behavior of the Life Cycle component instances. This table is populated and used in the State, Runtime, and Historical databases. Each row in this table corresponds to one (source or work) table <TABLE_NAME> that requires pruning. Altering column values for a given row only affects the Life Cycle component instance that has been assigned to prune table <TABLE_NAME>.

Column name	Column Type	Column Description	Nullable
TABLE_NAME .	CHARACTER	The fully qualified name of the table that is to be pruned.	N
LAST_PRUNED	TIMESTMP	The time of the last pruning operation on this table.	Y
LOGLEVEL	NUMERIC	The logging level that determines how much information will be put in the WBIRMADM.RMLOG table. Valid values are 0 and 1. 0 equals minimal logging; 1 equals maximum logging.	N
PRUNE_ENABLED	NUMERIC	A flag that determines whether pruning operations should occur. 0 indicates no, and 1 indicates yes.	N
PRUNE_INTERVAL	NUMERIC	The minimum amount of time in minutes between pruning operations.	N
RETENTION_IN_MINUTES	NUMERIC	The length of time in minutes after which an eligible row can be pruned.	N
ROWS_PRUNED	NUMERIC	The number of rows that were pruned during the last pruning operation.	N

Do not alter any of the following column values or unexpected behavior will occur:

- LAST_PRUNED
- ROWS_PRUNED
- TABLE_NAME

Changes to the following columns will be committed the next time an ETL component instance is invoked:

- LOGLEVEL
- PRUNE_ENABLED

- PRUNE_INTERVAL
- RETENTION_IN_MINUTES

Data movement service metadata and logging table

This section provides reference information about logging table structure in the WebSphere Business Monitor databases. The logging tables are static tables.

RMMETADATA

A varying number of component instances are used to provide data movement services for a given business measures model. Each one of the State, Runtime, and Historical databases contains a table that lists for each business measures model the names of component instances that have been assigned, as well as other useful internal information. This table is updated each time component instances are created and configured during the deployment phase. Do not modify its contents manually.

Column name	Column Type	Column Description
ID	NUMERIC	Not used
OM_NAME	CHARACTER	The name of the associated business measures model project served by these replication artifacts.
OM_ID	NUMERIC	Nnot used
MC_NAME	CHARACTER	The name of the associated business measures group served by these replication artifacts.
MC_ID	NUMERIC	not used
TGT_TAB_NAME	CHARACTER	The fully qualified name of the target table that is populated by the ETL stored procedure.
TGT_RM_APP_SVR_NAME	CHARACTER	The name of the server responsible for running replication apply operations.
TGT_RM_APP_SS_NAME	CHARACTER	If available, a group managed by the apply server, for DB2 SQL replication. This is a subscription set.
TGT_RM_APP_STG_TAB_NAME	CHARACTER	The fully qualified name of the staging table used as a target by the replication apply program. **Note:** There are two other tables that have the extension *_BKUP* and *_M* that exist in the system and are related to this row.
TGT_RM_APP_ERR_TAB_NAME	CHARACTER	The fully qualified name of the table that stores pointers to rows in the staging table that still need to be processed by ETL.

RMMETADATA

A varying number of component instances are used to provide data movement services for a given business measures model. Each one of the State, Runtime, and Historical databases contains a table that lists for each business measures model the names of component instances that have been assigned, as well as other useful internal information. This table is updated each time component instances are created and configured during the deployment phase. Do not modify its contents manually.

Column name	Column Type	Column Description
TGT_RM_APP_PRUNE_SP_NAME	CHARACTER	The fully qualified name of the stored procedure responsible for pruning the apply staging table on the target system.
TGT_RM_APP_TMP_TAB_NAME	CHARACTER	The fully qualified name of the temp table that is used by the ETL program to determine which rows should be loaded in the target table.
TGT_RM_SPETL_NAME	CHARACTER	The fully qualified name of the ETL stored procedure that is responsible for populating the target table from the entries in the staging table.
SRC_TAB_NAME	CHARACTER	The fully qualified name of the source table that is being replicated into the staging table.
SRC_RM_CAP_SVR_NAME	CHARACTER	The name of the server responsible for running replication capture operations.
SRC_RM_CAP_STG_TAB_NAME	CHARACTER	The fully qualified name of the table used by the Capture server to store changes to the source table.
SRC_RM_PRUNE_TRG_NAME	CHARACTER	The fully qualified name of the trigger responsible for removing selected rows from the source table during the Capture server pruning cycle. Selected rows may include rows representing completed operations.
SERVICE_NAME	CHARACTER	A label used to identify the service these artifacts belong to, for example, State_to_Runtime or Runtime_to_Historical.

Using the following simplified example view:

OM_NAME	SRC_TAB_NAME	SRC_RM_CAP_SV...	SRC_RM_CA...	TGT_RM_AP...	TGT_RM_AP...	TGT_TAB_NAME	SERVICE_NAME
STEW_S	wbi.CTX_TQ4MUF...	CAPTURE_1	CAP.CD_2	APPLY_4	APP.CCD_6	wbi.CTR_TQ4MUF...	State to Runtime
STEW_S	wbi.AI_BVSOYAP...	CAPTURE_1	CAP.CD_3	APPLY_4	APP.CCD_7	wbi.AIR_BVSOYA...	State to Runtime

It is easy to determine that source WBI.CTXTQ4MUF in the State database is being monitored by Capture component instance CAPTURE_1. Any changes to the source table are recorded in work table CAP.CD_2 and then applied by Apply component instance APPLY_4 to work table APP.CCD_6. This table is used by an ETL component instance to populate target table WBI.CTXTQ4MUF in the Runtime database.

RMLOG

The Runtime and Historical databases each contain a logging table that can be used to obtain statistics, progress, debugging, or error information. All ETL components and Target Life Cycle components write messages to this table but do not read from it. Some messages may be suppressed by setting the logging level to a minimum.

Column name	Column Type	Column Description
ENTRYSTMP	TIMESTAMP(10)	The timestamp for a particular entry in this log table.
ID	NUMERIC	An identifier to associate multiple rows from the same instance together. This ID comes from the SEQUENCE WBIRMADM.RMSPTRIGID.
ROWS_INSERTED	NUMERIC	An indicator of how many rows were inserted during this instance.
ROWS_UPDATED	NUMERIC	An indicator of how many rows were updated during this instance.
ROWS_DELETED	NUMERIC	An indicator of how many rows were deleted during this instance.
ROWS_INERROR	NUMERIC	An indicator of how many rows were marked as causing a recoverable error during this instance.
NAME	CHARACTER	The fully qualified name of the stored procedure, trigger, or process that caused the entry in this table.
OPERATION	CHARACTER	A label identifying the operation that was being performed when this entry was made.
RESULT	CHARACTER	A column where more information could be found about the operation that occurred.
ISTRACEENTRY	NUMERIC	A column indicating whether this entry requires LOGLEVEL (in WBIRMADM.RMCONTROL) to be set to 1. 0: This log entry is not a trace entry. 1: This log entry is a trace entry (and may be suppressed - see WBI.RMCONTROL table).

Each row in this table corresponds to a message that was issued by component instance <NAME> at <ENTRYSTMP> . Rows having the same <ID> and <NAME> represent messages that have been generated during the same invocation of <NAME>. The following example contains log entries that were generated by ETL component instances WBIRMADM.WBIRMSP_10 and WBIRMADM.WBIRMSP_14 as well as Target Life Cycle component instances WBIRMADM.WBIRMSP_P13 and WBIRMADM.WBIRMSP_P_17. WBIRMADM.WBIRMSP_10 (4:40:20 PM) and WBIRMADM.WBIRMSP_14 (4:40:27 PM) issued five messages each and WBIRMADM.WBIRMSP_P_13 (4:40:20 PM) and WBIRMADM.WBIRMSP_P_17 (4:40:20 PM) one.

ENTRYSTMP	ID	NAME	OPERATION	ROWS_INSERTED
Oct 11, 2005 4:40:20 PM 3...	1	WBIRMADM.WBIRMSP_10	SP_START	0
Oct 11, 2005 4:40:20 PM 3...	1	WBIRMADM.WBIRMSP_10	DEL_TEMP	0
Oct 11, 2005 4:40:20 PM 3...	1	WBIRMADM.WBIRMSP_10	INS_TEMP	0
Oct 11, 2005 4:40:20 PM 3...	1	WBIRMADM.WBIRMSP_10	FETCH_TARGET_...	0
Oct 11, 2005 4:40:20 PM 3...	1	WBIRMADM.WBIRMSP_10	SP_END	0
Oct 11, 2005 4:40:20 PM 3...	2	WBIRMADM.WBIRMSP_P_13	PRUNESTAGING	0
Oct 11, 2005 4:40:20 PM 3...	3	WBIRMADM.WBIRMSP_P_17	PRUNESTAGING	0
Oct 11, 2005 4:40:27 PM 1...	4	WBIRMADM.WBIRMSP_14	SP_START	0
Oct 11, 2005 4:40:27 PM 1...	4	WBIRMADM.WBIRMSP_14	DEL_TEMP	0
Oct 11, 2005 4:40:27 PM 1...	4	WBIRMADM.WBIRMSP_14	INS_TEMP	0
Oct 11, 2005 4:40:27 PM 1...	4	WBIRMADM.WBIRMSP_14	FETCH_TARGET_...	0
Oct 11, 2005 4:40:27 PM 1...	4	WBIRMADM.WBIRMSP_14	SP_END	0

This table is not automatically pruned. The DBA should regularly monitor and prune it. Use the information found in WBIRMADM.RMMETADATA to identify for which business measures model the component instance <NAME> provides service. Note that the value of the LOGLEVEL and ETL_0_MINUTES columns from the WBIRMADM.RMCONTROL table and the value of the LOGLEVEL and PRUNE_INTERVAL from the WBIRMADM.RMPRUNECTRL table will affect the growth rate of this table. More entries will be made when LOGLEVEL is set to 1, when ETL_0_MINUTES decreases, and when PRUNE_INTERVAL decreases.

Database services troubleshooting

During generation, deployment or while running the database services of WebSphere Business Monitor, it is possible that errors related to database services may occur. The following is information on how to troubleshoot database related problems.

Deployment issues

During the different deployment scenarios of WebSphere Business Monitor databases artifacts, errors may occur. The following are the proposed solutions for each error.

During deployment of the generated database artifacts problems can arise that are a result of

- Incorrect configuration
- Insufficient user privileges
- Environment setup problems

Table 2. Deployment errors

Problem	Solution
Table space assigned to a table does not seem to exist.	• Make sure the table spaces defined in the table-space properties file exist with the characteristics described. • Either create the missing table space, with the appropriate characteristics and rerun the DDLs, or update the table-space properties file to match the table spaces that are defined, and then regenerate the schema.
Table space assigned to table is too small to hold the table.	• Make sure the table spaces defined in the table-space properties file exist with the characteristics described. • Either fix and rerun, or just edit the DDL manually to correct the table-space assignment.

Table 2. Deployment errors (continued)

Problem	Solution
Tables already exist in the database.	Assuming this DDL has not been run previously, there are two possible causes of this problem. • The DDL scripts were generated with the option **Ignore Previous Deployments** selected. The Schema Generator generates new table-creation statements, rather than altering existing tables. This option should only be used when you are trying to create your database tables initially; that is, when you have dropped your existing database tables. If you know you have existing database tables and want to retain them, rerun the Schema Generator without the **Ignore Previous Deployments** option selected, and then rerun the resulting DDL scripts. • The other cause is that at some point a version of this business measures model has been removed and the *Delete and keep for reporting* option was not selected. If a version of an business measures model is deleted and the *keep for reporting* option is not selected, then the Schema Generator cannot continue to manage changes in the supporting database tables for this business measures model. There are two options at this point. – Branch off and create a new business measures model based on the current one, and deploy it as a new business measures model with a new set of tables. You can manually migrate the data from the existing set of tables into the new tables. – Manually drop the existing database tables that support this business measures model, using the provided mapping files as a guide. Once the tables have been dropped, rerun the Schema Generator with the **Ignore Previous Deployments** option. The resulting DDL scripts will create a fresh set of tables that will support this latest version of the business measures model. **Note:** Unless you manually back up the data from the old tables before you drop them and then migrate the data into the newly created tables, you will not have a reporting history for any processes that ran on the old versions of the business measures model. is not recommended to use the *delete* option instead of the *delete and keep for reporting* option when you are removing the model, unless you have no intention of deploying a new version of this business measures model at some point in the future.
Table space is too small. (Although the table space initially assigned to this table was sufficient for the column size, subsequent metrics that were added to the table have pushed it beyond the page size of the current table space.)	You will need to back up this table, drop it, and then re-create it, assigning it to a larger table space. The backed-up data should then be loaded into the new table. Once the current table is re-created in a larger table space, you will be able to run the latest DDL scripts that will add the necessary columns.

For various reasons, you may want begin again with your database tables. For example, you may have a number of metrics that you no longer want, which are still in your database tables because they existed in previous business measures model versions. The simplest way is to rename the project in WebSphere Business Modeler. The business measures model will be treated as a new mode, and new, unique tables will be created in the database.

Note: In this case, no history information from previous process instances will be available.

If you want to view the history information, copy the data from the original database tables into the newly created tables. The column names will not match, but the data types will be matching. You can use the mapping files (generated with the DDL scripts) or database column comments to identify which column corresponds with which metric and which table corresponds with which process.

Note: Some columns from the original tables will not have corresponding columns in the new tables if those metrics no longer exist in the latest version of your business measures model.

Runtime issues

While you are restarting a Capture server that had been down for several days, you receive an error message produced by the Capture server from IBM DB2 replication. The message appears in a Capture window of a Windows system, in a log file on the system, e-mailed as part of a replication monitor transmission, or in a IBMSNAP_CAPTRACE table.

Error message

ASN0121E CAPTURE "CAPTURE_141" : "WorkerThread". The Capture program warm start failed because existing data is too old. The Capture program will terminate.

To solve this error, refer to Preparation of database artifacts deployment

Stopping Runtime database

When stopping or bringing down the WebSphere Business Monitor Runtime database for any reason, you should first stop the adaptive action manager application.

You can stop the adaptive action manager application from the WebSphere Process Server Admin Console. The WebSphere Business Monitor Runtime database should be started first before starting adaptive action manager.

Notices and Trademarks

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Lab Director
IBM RTP Laboratory
3039 Cornwallis Road
P.O. BOX 12195
Raleigh, NC 27709-2195
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not necessarily tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

COPYRIGHT LICENSE

This information may contain sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries, or both:

IBM
IBM (logo)
WebSphere
DB2
Tivoli
MQSeries
AIX
z/OS

Excel, Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, MMX, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

ALPHABLOX is a registered trademark of Alphablox Corporation in the United States, other countries, or both.

Adobe is trademark of Adobe Systems Incorporated in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.