CA Business Service Insight

Implementation Guide

8.2.5

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Introduction

This guide discusses the implementation process and related roles, responsibilities, and interactions involved in planning, design, implementation, installation, and deployment of CA Business Service Insight.

This section contains the following topics:

Roles

A roles is a person who performs a common set of tasks that are performed during a typical implementation. The word role can also refer to the set of tasks themselves. CA Business Service Insight has a number of predefined roles which can be edited by the system administrator. In addition, new roles and specific permissions can be created.

The following roles are required for implementation:

- Service Catalog Manager
- Contract Manager
- Business Logic Expert
- Data Sources Expert
- System Administrator

The responsibilities and qualifications of each role are described below.

Note: The same person can perform several roles, as defined by the System Administrator.

Service Catalog Manager

Responsibilities:

- Manage the organizational service catalog
- Define the organizational service offering catalog
- Manage requirements for contract definitions and reporting

Required qualifications:

- Familiarity with the organization's E2E service delivery process
- Familiarity with CA Business Service Insight concepts

Contract Manager

Responsibilities:

- Interaction with the end user
- Being in charge of building the Service catalog based on defined requirements
- Create new contracts, maintaining existing contracts
- (Optional, recommended) Being in charge of specific customer SLA Implementation

Qualifications:

- Understanding principles and logic of metrics
- Expert user of CA Business Service Insight GUI-based features

Business Logic Expert

Responsibilities:

- Implement metrics
- Write business logic scripts; maintain existing business logic scripts

Qualifications:

- Basic development background, strong working knowledge of scripting languages such as VB-Script
- Good understanding of CA Business Service Insight data flow
- Expert in CA Business Service Insight Business Logic scripting
- Good understanding of CA Business Service Insight architecture and components
- Expert user of CA Business Service Insight GUI-based features

Data Sources Expert

Responsibilities:

- CA Business Service Insight Infrastructure design
- Create new adapters; maintain existing adapters, interface with operational infrastructure to obtain measurements
- Build and maintain CA Business Service Insight infrastructure

Qualifications:

- Understand customer data sources environment and structure
- Good working knowledge of database, XML, and scripting languages
- Understand CA Business Service Insight data flow and data gathering processes
- Expert in CA Business Service Insight Adapters
- Understand CA Business Service Insight architecture and components
- Expert user of CA Business Service Insight GUI-based features
- Development background an advantage

System Administrator

Responsibilities:

- Manage installation and upgrades
- Fulfill hardware and software system requirements
- Manage security settings: User and Role definitions
- Monitor and manage the system

Qualifications:

- Understand customer infrastructures and networks
- Understand CA Business Service Insight architecture and components
- Working knowledge of DB, XML, and scripting languages
- Understand CA Business Service Insight data flow
- User of CA Business Service Insight GUI-based features

Solution Process

The solution process includes three basic stages:

- Planning and Design
- Implementation
- Installation and Deployment

The roles described previously participate in different stages of the implementation process, each according to their specialties. These roles interact to complete the whole process from beginning to end, from contractual definition to output report.

This guide is structured as role and process oriented, based on a general stages flow that builds the implementation process. It describes how each role is involved and how the roles interact.

The following paragraphs provide an overview of the stages included in the implementation process and the activities of each role.

The rest of the guide is structured by stage with an indication which roles participate in the stage. Each chapter indicates for each activity the role responsible for it.

Following is a brief description of the basic tasks of each phase and the functions of the different roles. Additional details can be found within each chapter.

Planning

The purpose of the planning stage is to identify and quantify all aspects that must be addressed as part of the implementation.

At this stage, the Contract Manager collects business requirements from the Service Catalog Manager to understand the expectations from CA Business Service Insight following a successful implementation. It is important at this stage to understand both the current requirements and future plans to verify that the implementation provides easy and smooth growth and evolution.

According to the defined implementation requirements, the Contract Manager is required to examine the existing related process (if it exists) by reviewing the existing process inputs and outputs. It is necessary at this stage to identify all required information sources and to obtain samples, contracts, output reports, and input sources used to calculate the existing outputs. This information must be carefully specified for the Contract Manager to identify all input needed, so that the expected output can be derived.

At this stage, the Contract Manager examines the Contracts to verify that the Service Catalog and Templates provided as part of the implementation support existing and future needs, and that the current implementation covers all contract areas.

The Contract Manager examines the reports and their formats to verify that all defined measurements are done to be able to produce them with the existing implementation.

The Data Source Expert then examines input data samples per required measurements and calculations provided by the Contract Manager. This allows the Data Source Expert to identify any input issues that need to be addressed, such as custom formats or deficiencies, and determine whether additional data sources are required.

The purpose of this examination is to verify that the sources contain the required information needed to calculate the required Metrics and initial information about the retrieval process, such as methods of communication with the data-sources and data-source structure.

Design

During the design phase, input and output gathered requirements are translated into the CA Business Service Insight model structure comprising Contracts, Metrics, and Resources. This means transforming real-world data and conceptualizing it so that it fits the CA Business Service Insight framework.

The system design includes the following:

Contract modeling

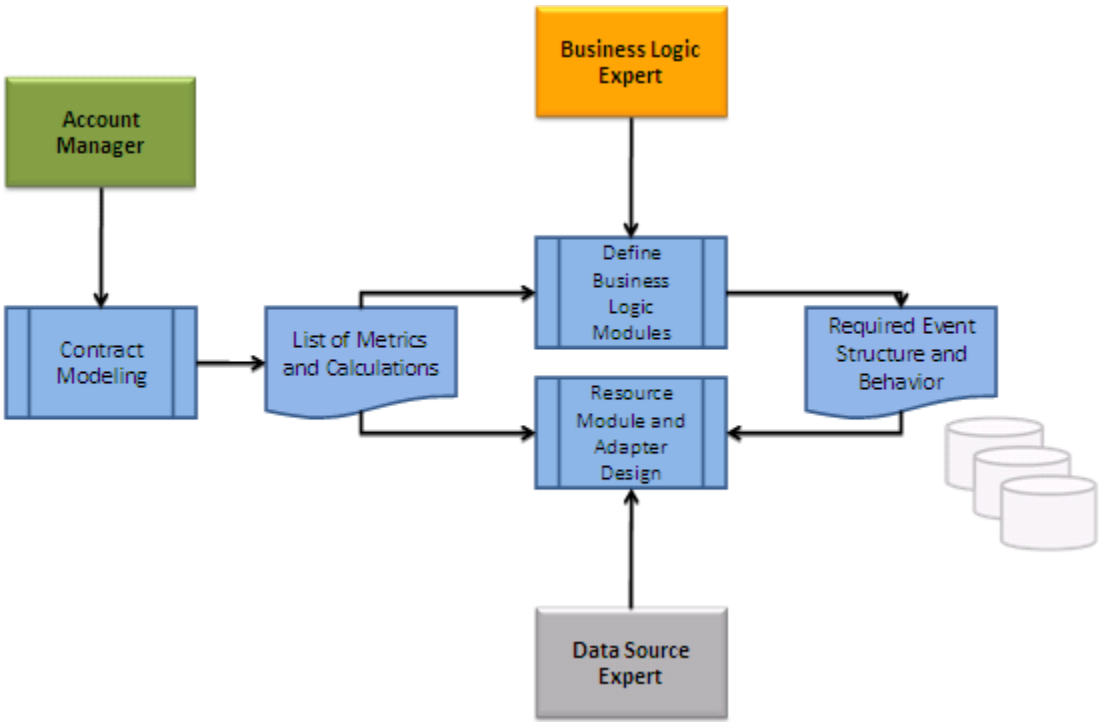
Customer SLAs are interpreted into CA Business Service Insight contracts and service catalog entities (such as the Template Folders) are defined. This is done mainly by the Contract Manager.

Data modeling

Resource data is examined and modeled into the CA Business Service Insight resource model. This is done by the Data Source Expert and Business Logic Expert.

There may be several methods available for either contract or data modeling. However, often there is an optimal practice that not only solves the problem, but also offers more flexibility or productivity, thereby providing a strong framework for further growth.

To assist the Data Source Expert in choosing the most suitable methods, case studies are used with suggested solutions.



As shown in the previous workflow diagram, as a result of the Contract modeling process, the Contract Manager provides as an output the list of Metrics that should be configured in the system and their calculation outline definition.

Example:

Customer A, CNP System average response time.

The Metrics list is provided as an input for the Business Logic Expert that defines the required event structure and event behavior that allows defining the required calculation script.

The Metrics list and the event structure and behavior requirements are the input for the design of the resource model and the data adapters by the Data Source Expert.

Implementation

At the Implementation phase, the CA Business Service Insight system is configured based on the results of the design phase. The Implementation phase involves taking the theoretical design stage results and using them to build the operational requirements for CA Business Service Insight.

Some of these items to be created or addressed include:

Contract Manager

- Service setup
- Contract creation
- Reports and Dashboard setup

Data Source Expert

- Adapter configuration
- Infrastructure setup

Business Logic Expert

- Business Logic Modules

Installation and Deployment

The installation and deployment phase is concerned with the installation and integration of the production system, testing and monitoring its performance, and training the users. The System Administrator and the Data Source Expert perfom most of the activities at this stage.

Chapter 2: Planning and Design

This section contains the following topics:

Requirement Gathering Session (All Roles)

The requirement gathering session is the first critical step in the implementation of CA Business Service Insight, and should include all key personnel involved in the implementation.

The information required for this session must be provided by people who are very familiar with the selected SLAs, including the SLA business logic, how the SLAs are currently being managed, what reports are currently being generated for them, and what full reporting requirements are expected in the future.

The following personnel must participate:

- Service Catalog Manager(s)
- Contract Manager(s)
- Data Source Expert(s)
- Business Logic/Scripting Expert(s)
- Any other relevant staff members familiar with the selected SLAs
- Project manager (if appointed)

CA recommends the following:

- It is important to ensure there are personnel present who can make decisions regarding any unclear definitions that might arise while reviewing selected SLAs.
- It is advisable that the implementation team receive, prior to this meeting, a copy of the SLAs selected for this project. The attendees should be familiar with all relevant data sources that pertain to the selected SLAs, be able to provide data dumps of them, and explain their inner structures, where applicable.

The main goals of the planning session are to define and specify:

- Scope of planned work
- Success criteria
- Roles and responsibilities of all involved parties
- Implementation process and schedule
- Required hardware/software requirements

This is accomplished by:

- Reviewing SLAs to be used in implementation
- Reviewing "Business Logic" for each objective within selected SLAs
- Reviewing reports, alerts, and Dashboard requirements related to selected SLAs
- Reviewing relevant data sources to be used in implementation

- Reviewing relevant Infrastructure topology
- Collecting relevant data dumps from relevant data sources
- Identifying key people and their responsibilities
- Defining communication paths for duration of implementation, review meetings, Q&A process, and so on
- Reviewing implementation's process and schedule
- Reviewing current state of affairs, that is, how are these SLAs managed, and what is missing
- Defining implementation's success criteria
- Defining required hardware/software requirements
- Defining time line for implementation

Collecting Sample Data (Data Source Expert)

To perform initial configuration of the Adapters off-site, it is important for data gathering purposes to obtain samples of past data. These samples must be of the same structure as that of the data that later appear incoming from the actual system from which CA Business Service Insight needs to retrieve data. In addition to the samples, the Data Source Expert should learn about the data source from the data source owner in order to be able to clarify the following issues:

Type:

Database, file, stream or other.

Location:

Where is it? How to get there (connection details)? Security obstacles? Platform?

Naming convention:

How are files named? (is it a file-based data source) How are tables named? Table fields names?

Availability:

When can it be accessed? When should it be accessed (load considerations)? Continuous update or once per X time? For how long is it persistent?

Behavior:

Is data accumulating? Is data overwritten? Is an archive kept?

Span:

How much historical data is available?

Volumes:

Current volume of data? Growth rate? Predictions?

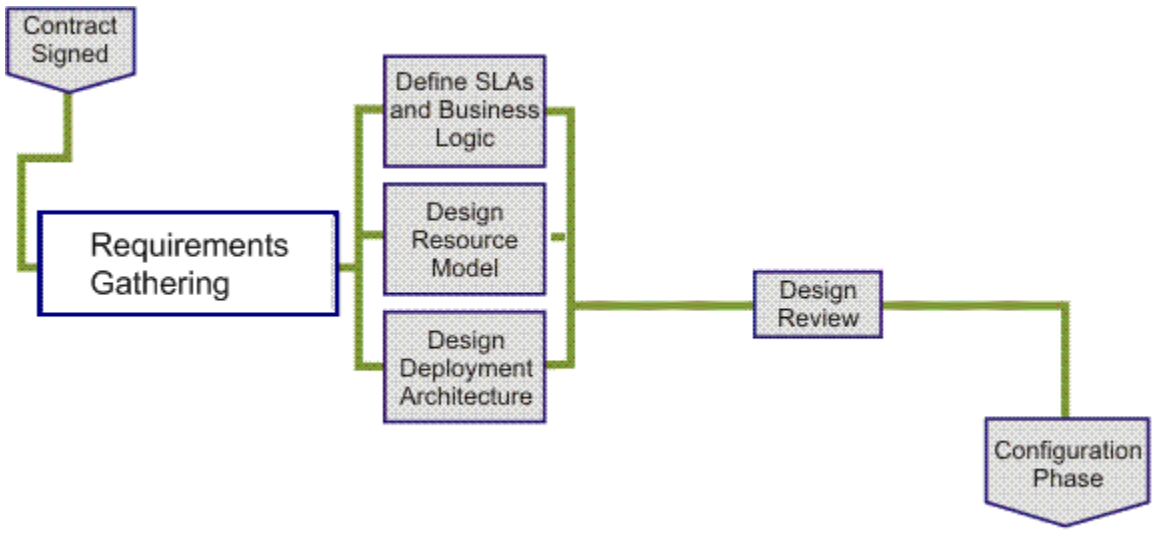
Structure and format:

How is data organized within data source? What are the data fields? What are the table names? What separates data fields?

Streaming:

Does the adapter get data or data is gathered by the Adapter?

Design


Contract Signed
Requirements Gathering
Define SLAs and Business Logic
Design Resource Model
Design Deployment Architecture
Design Review
Configuration Phase

Design Introduction (Contract Manager, Business Logic Expert, Data Source Expert)

This section explains the process and reasoning behind the design phase of the solution process. The design phase follows the planning phase and is subsequently followed by the implementation phase in the following chapter.

The objective of the design phase is for the implementation team to be able to complete mapping the real-world contracts and their clauses, and existing performance data into the CA Business Service Insight system.

Before beginning this process, the implementation team needs to be aware of the various methods and options available so as to ensure that not only are all requirements taken into account, but also that the resulting design is as optimized as possible, while at the same time allowing for future growth or change.

The design process involves the Implementation team performing the following steps:

- Examine the contracts and convert them into the necessary CA Business Service Insight objects, referred to in this guide as "Contract Modeling"; responsibility of the Contract Manager.
- Take existing data sets and decide what and how to extract the relevant items in light of the desired results, referred to in this guide as "Data modeling"; responsibility of the Data Source Expert and the Business Logic Expert.

The Contract Modeling section explains:

- Terminology used (essential in correct implementation)
- Business Logic templates and modules
- Service Level Template and how to use them
- Importance of creating a strong service catalog
- Financial management Metrics (Penalties, Incentives and Costs and examples) as applied to customer contracts and other contractual Metrics

The Data Modeling section explains:

- Events as applied to CA Business Service Insight and their flow through the CA Business Service Insight system
- Metrics and how they are registered
- CA Business Service Insight Resources and how to identify them
- Suggestions how to automate the collection and definition of these Resources

All of the above points are explained in further depth in the following sections.

It is important to be aware that poor choices made at this stage can adversely impact the operation of CA Business Service Insight and may be difficult or impossible to reverse at a later stage.

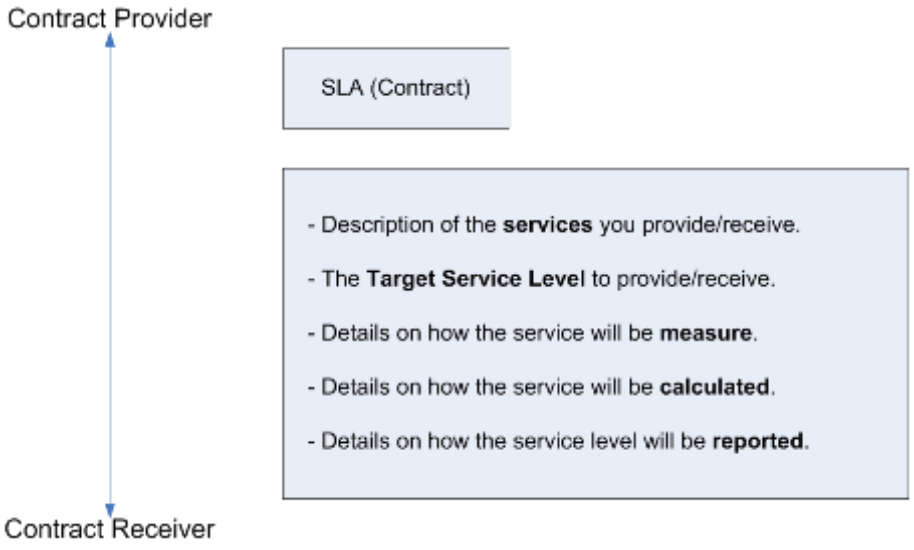
Contract Modeling (Contract Manager)

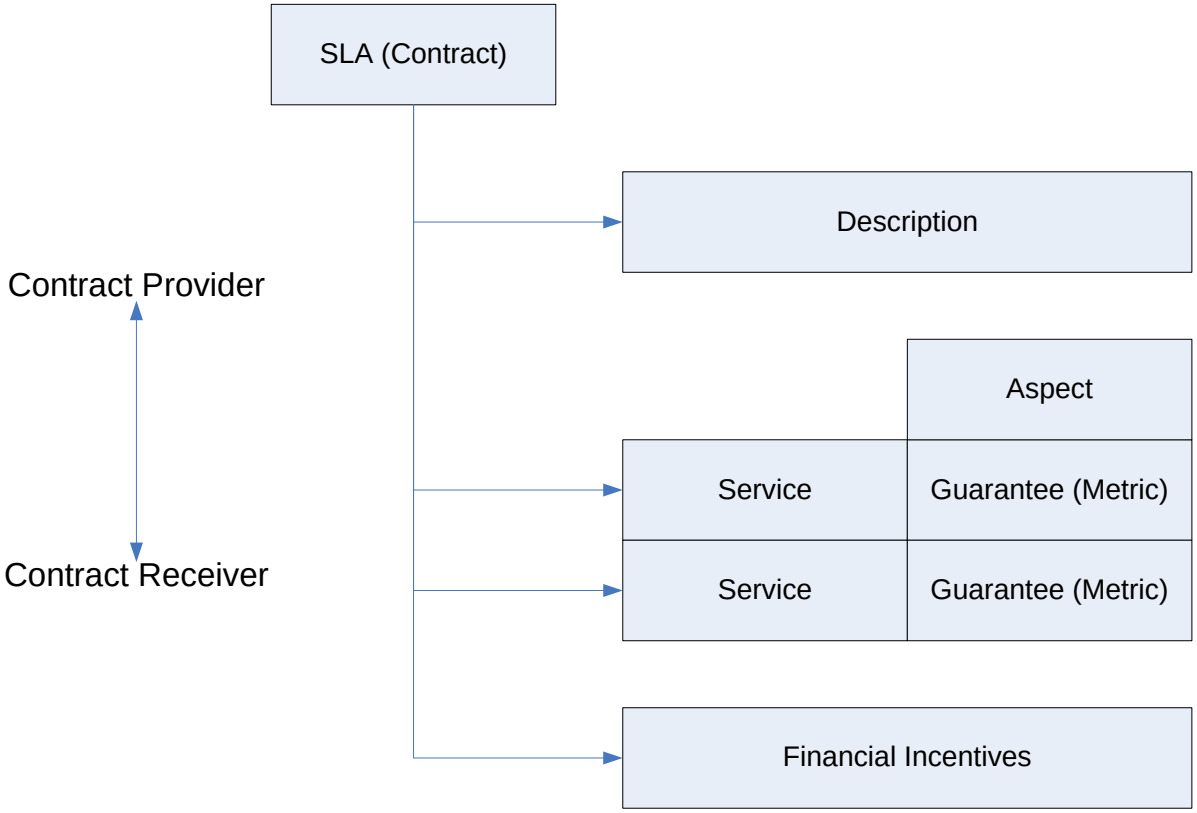
The following tasks for contract modeling are carried out by the contract manager.

Contract Terminology

A Contract is a collection of objectives. An objective is either a contractual or operational objective (Metric) or a financial objective (Incentive). The process of Contract Modeling involves taking the entire Contract content that lies within scope of the implementation and converting it into the CA Business Service Insight model.

The following diagrams depict this process.

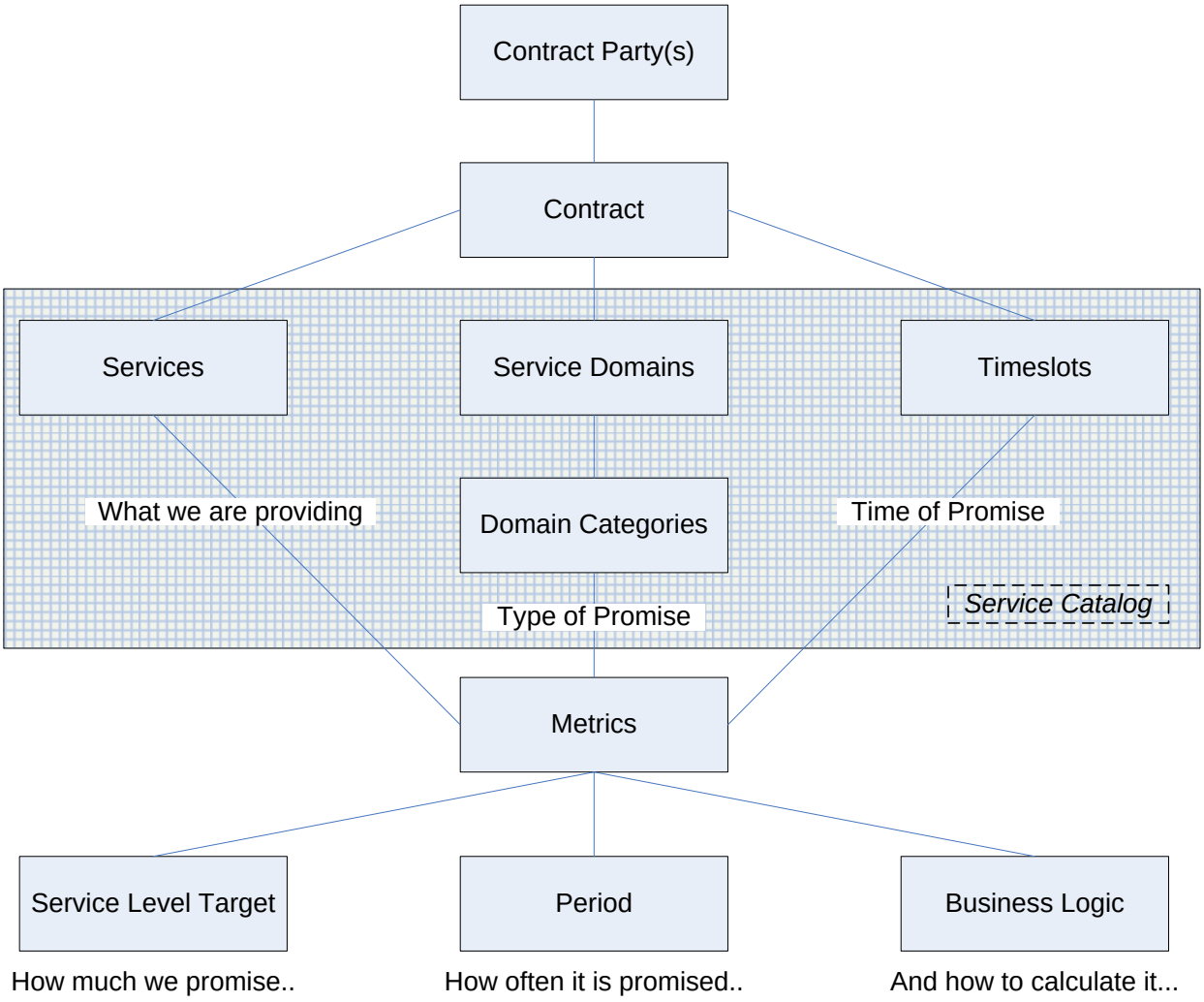




Note: Possible future plans for the system must be taken into account even though only aspects which fall within the implementation scope will be modeled. The scope must include general Contract management requirements of the organization in order to design a broad enough model that encompasses future developments. Doing so minimizes and simplifies any changes that need to be made in the future.

Converting customer contracts into the CA Business Service Insight model is a process whereby the Contract Manager groups the guarantees (Metrics) offered within common bounds into logical groups, common service components, Contract aspects, and so on. This logical grouping allows for very flexible service level management.

The CA Business Service Insight Contract model comprises the entities shown in the following figure:



Contract

Defines the agreement and collection of Metrics. A contract can be one of three different types depending on the relationship between the Contract Parties involved. It can be either an SLA (Service Level Agreement between the organization and its customers), OLA (Operational Level Agreement between divisions within the organization) or a UC (Underpinning Contract between the organization and its suppliers, where generally the UC is on a service for which the organization is supplying to another customer via an SLA).

Contract Party(s)

Defines the customer of the services provided, and with whom they signed the contract. A single Contract Party can be attached to more then one Contract. Note that within the contract you are able to define the provider and receiver of the services pertaining to that contract.

Metrics

Defines a single service level objective or guarantee. Each Metric is a request for measurement that produces the actual service level result on which reporting is performed.

A Metric comprises the following items:

Type

A Metric can be one of the following eight types:

SLO	Service Level Objective.
Informational	Service Level Objective without a target.
KPI	Key Performance Indicator, used to support different industrial concepts. In Telco KPI=SLO and means customer obligation while in IT it means a technical obligation.
KQI	Key Quality Indicator, an overall metric to measure aggregated result from different indicators.
Interim	Interim measurement to use in other metrics. These metric results cannot be reported.
Consumption	Used by Financial metrics to calculate service/resource consumption. Used in conjunction with Price Item metrics
Incentive	Financial metric, used for incentive (positive term for penalty) calculations.
Price Item	Financial metric, used to calculate results based on consumptions. Different prices per period or per number of units.

Tracking Period

Contractual reporting period or period for which the actual calculated service level result is compared with the target. The tracking period in CA Business Service Insight can be defined with a granularity of Hour, Day, Week, Month, Quarter, or Year. For the purposes of root cause analysis, in addition to the tracking period, the metric is also calculated in the other six granularities, but the results for these periods are marked as operational results only.

Note: This is only performed if these calculation granularities are specified for the metric.

Timeslot

Time within the tracking period during which the specific guarantee or obligation is applicable, including exceptional timeslot definitions, such as predicted maintenance windows, standard bank holidays, and so on.

Business Logic

Formula that defines how to evaluate raw data in order to calculate the actual service level value for that aspect of the service, and the specific assumptions the calculator needs to take into account. During the design phase, only the outline of the calculation is identified. It is defined in greater detail during the Configuration phase.

Target

Contractual service level obligation. The target may or may not be mandatory, depending on the type of metric in question. In the case where a target is not defined, the metric provides results for reporting purposes only, and it lacks the ability to be compared with a contractual obligation or guarantee. Targets can be static or dynamic.

Note: To follow some of the concepts presented so far, refer to Appendix B - Case Studies (see page 213).

Each Metric is related to the following system-wide entities contained within the system template folder:

Service Component

Describes what is obligated to provide.

Service Component Group

Collection of Service Components. A single Service Component may be a part of more than one group. A Service Component Group is an optional entity, if used, can provide a greater flexibility for reporting.

Service Domain

Specific aspect of service components that need to be measured for the purposes of service level monitoring (for example, Performance or Availability).

Domain Category

Specific unit of measurement. This defines the default unit of measurement and whether the target objective is a minimum or a maximum value. Essentially this is the specific dimension of the service domain being measured (that is, % of time available, number of outages, average throughput rate, and so on).

Each of the above entities, as well as their relationships to all service level objectives that are to be monitored in CA Business Service Insight, need to be identified during the Contract modeling stage.

How the Modeling Process Works

During the Modeling process all Metrics that need to be configured in the system must be defined together with their related entities, based on the Contract under consideration and the reporting requirements.

Before or during this stage it is good practice to decide upon a naming standard to use for the Metric names so that the system looks neat and tidy and is easy to navigate. Also consider the description of the Metric that can be used within the Objective Statement section of the Metric.

The contract analysis process includes the following steps:

1. Identifying the contractual objectives.

   For each objective, identify:

   - A suitable name (keep in mind any naming standards)
   - Target (optional)
   - Tracking period
   - Measurement Unit
   - Service Component
   - Service Domain
   - Domain Category (and measurement unit)
   - Timeslot (when: 24 x 7, business hours only?)
   - Calculation outline (what variables are needed and how is the service level calculated)

   Note: Some of those may not be clear from the first pass through, but can be sorted out later when refining the service catalog.

2. Identifying, from the Contract, any financial objectives and determining for each financial objective:
   - Is it a penalty, incentive, or cost objective?
   - What are the conditions that trigger it?
   - For which Service Component(s) does it occur?
   - Service Domain
   - Domain Category (the unit here may be a currency or cost amount or a percentage of payment, and so on)

Once all the objectives have been noted and defined, it is recommended to review the complete list of Metrics and try to optimize/normalize the catalog.

During this step it is important to make sure that each of the Service Components, Service Domains, and Domain Categories are defined sensibly and that they can form a clear and concise catalog of what is offered throughout the system. This includes ALL of the Metrics and Contracts in the system. This ensures building a very strong service catalog within the system to enable future growth of the system.

Service Domains and Domain Categories should NOT be defined down to a very low granular level. They should be descriptive but at a higher level than the metric.

For example, if you have the three following Metrics:

- % of outage reports delivered on time
- % of exception reports delivered on time
- % of management reports delivered on time

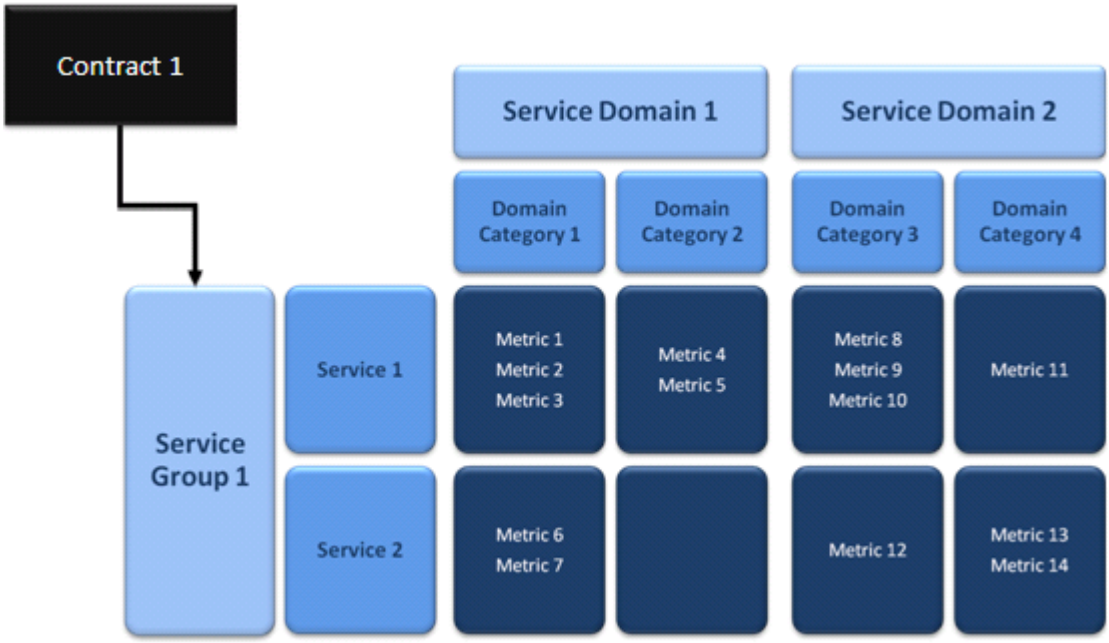
The best category all three Metrics would probably fall under is the service domain of Performance and the domain category of % Reports delivered on Time. The Domain Category should not mention the type of reports. These three Metrics would likely have the same calculation method and use the same business logic (with the exception of maybe a single parameter to distinguish between the different report types).

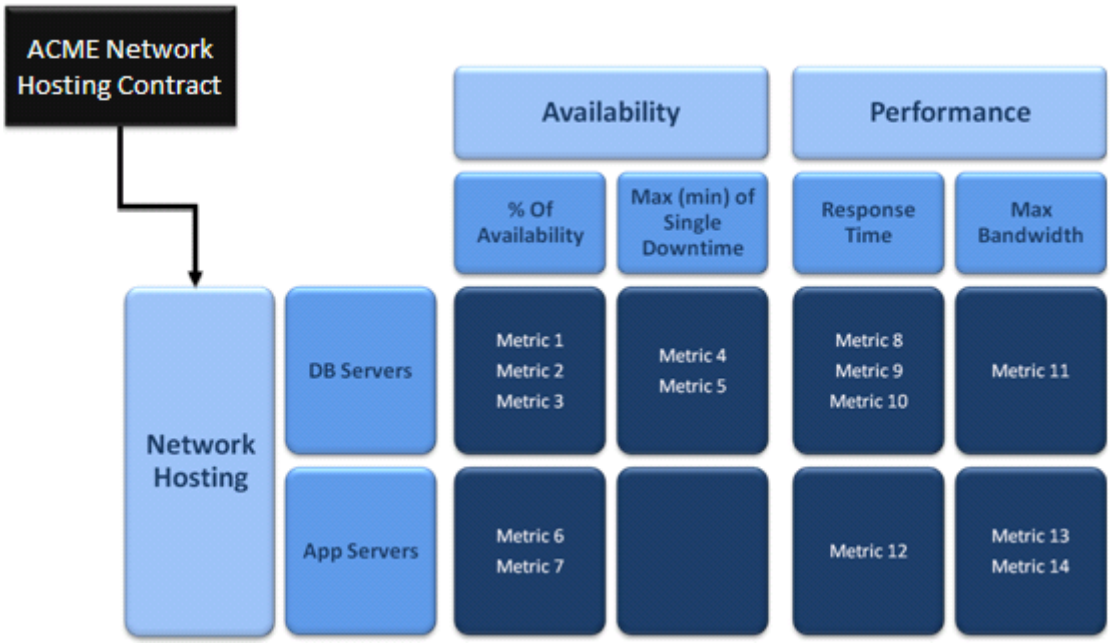
Suitable Service Domains and Domain Categories can be taken from the ITIL (Information Technology Infrastructure Library) standard. These are only suggested examples, however, and each particular organization may have their own defined standards for these. Refer to Appendix A - Sample Service Domains and Domain Categories (see page 211) for some suggested service domains and domain categories.

Note: It may be useful at this point to hold a meeting with all key stakeholders, in order to get the buy-in from each of them that the chosen catalog supports their current and future needs.

Further examples that demonstrate important points are presented in the appendices. In these examples, a single Contract objective is described along with its modeling. When modeling actual situations, it is necessary to be cognizant of all of the objectives so that the Catalog Entities represent all the objectives in a broader and all encompassing fashion.

Once the process of identifying all Metrics and their related entities is complete, the Contract Manager has a matrix that describes all of the Metrics as shown in the following diagram.





Additional issues to be considered in the Modeling process are described in the following sections.

Questions for the Contract Manager

The following are the questions the Contract Manager must consider in order to ensure that all aspects have been taken into account:

How do I know I've chosen the correct Service Component?

> If the service component can be applied to more than one Contract and can be measured in more than one aspect, it has been defined correctly.
>
> For example, System X is a system provided to more than a single customer and can be measured by its availability, reliability, performance, and so on.

How do I know I've chosen the correct Service Domain?

> If the Service Domain can be measured or calculated in more than one way, it is describing a general aspect of the service provided and is therefore the correct Service Domain.
>
> For example, the availability can be measured in number of ways, one of which is the percentage of time available. Others can be the percentage of availability during or out of business hours, the number of failures, mean time between failures (MTBF), mean time to repair, maximum down time, average down time, total down time, and so on. All of these are different ways with which to evaluate the availability of a particular system.

Cases to be Considered During Modeling Process

Following are several examples for cases some common and some more unique that describe concepts that should be taken in to account in the modeling process. These concepts may result with a more precise definition of the metrics and a stable framework.

Target-less Metrics

Since the target field within the Metric definition is not mandatory, in cases where it is not defined, service-level reports are available for the Metric. However, no Service Level versus Target and Deviation reports are possible (since there is no target with which to compare the actual calculated service level). These types of Metrics are defined in cases where reports are required for information that is not part of actual contractual obligations.

Defining this type of Metric provides the user with all of the possible drill-down capabilities of reporting, as well as providing the Service Level Manager with the option of applying the measurements to a target at any stage in the future.

***For Example*:**

The contractual guarantee is to provide 99% of network availability and report on the number of down times per each month.

Two Metrics are defined, one Metric is defined with a target of 99% for availability, and another Metric is defined for counting the number of down times for each month without a target. Both Metrics are reportable, but only the first has deviation calculations due to its contractual obligation.

Note: Another possible method for addressing this kind of situation is to use Business Logic outputs and free-form reporting on this data. However, this loses the drill-down capability of the report on the data as well as the option of using the simple report wizard. The advantage of using Business Logic outputs, on the other hand, is one of saving Engine power by having fewer Metrics.

For further information about this method refer to the Outputs-User Tables (see page 152) case study.

Metrics with Targets

In the cases where a target is defined for a Metric, there are two possible ways of specifying the target. It can be specified as either a static or dynamic target. A static target is the most common scenario encountered, where the target can be an agreed-upon value valid for the duration of the contract.

For example:

Network availability should be no less than 98% every month.

The target in this case is 98%.

Alternatively, the target can depend on the previous months' performance, or just change its value throughout the year. There are many alternative situations which could be encountered here, but in general they are all implemented by way of a formula. CA Business Service Insight supports this feature by an additional function call from the standard Business Logic template. The target function can access other parameters from the context of the Business Logic and can support any possible scenario required.

For example, resolution time of tickets in the Helpdesk that depends on the Helpdesk load: The average resolution time for high priority tickets is 1 day if there are no more then 1000 tickets during the same month. If there are more then 1000 tickets issued in the helpdesk within the month, the average resolution time for high priority tickets is 2 days.

In this case, the Metric is defined as having a dynamic target that is evaluated within the Business Logic script according to the number of tickets issued that month.

Note: For details on the implementation method of dynamic target refer to the Implementing Dynamic Targets case study.

Metric Parameter

A Metric parameter is a value that can be addressed from within the Business Logic of the Metric and one which can be easily changed from the Metric definition without needing to change the actual code. It is used in place of the hard-coded value and can be easily changed.

It is important to identify metric parameters to easily identify the Business Logic modules and to create reusable content. In addition, Metric Parameters are accessible through the Contract Wizard which allows an end user to perform changes easily.

***For example*:**

- Severity 1 incidents should be resolved within 24 hours.

  In the above obligation the target is a resolution rate of 24 hours and the severity level (Severity1) can be defined as a parameter.

- Number of down times during a month should not exceed 3 times, where down time is the time the system was not available for more then 5 minutes.

  In the above obligation, the time that is considered to be defined as down time can be defined as a parameter.

Contract Parameter

A Contract parameter is a value that can be addressed by all Metrics within a Contract. A Contract parameter can be used within the Metric using the same method as a Metric parameter but is instead defined as a dynamic parameter.

It is recommended to use a Contract Parameter when more then one metric requires using the same value. Another important incentive to use a Contract Parameter is for facilitating contract maintenance. Since parameters tend to change often and require updating within the system, it is easier to access a single location in the contract and change all parameters values concurrently than to access each metric in the contract and change the values of the parameters at the metric level.

Therefore, the most recommended modeling is to define the parameters in the contract level as Contract Parameters and to access their values through the Metric level dynamic parameters.

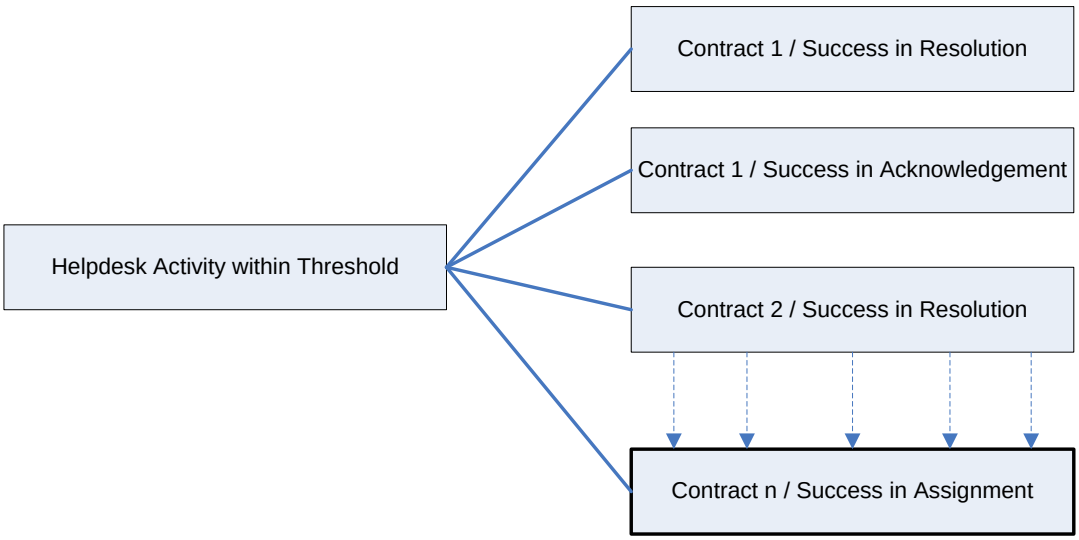
For an example, refer to the Helpdesk Performance (see page 221) case study.

Business Logic Templates and Modules

Business Logic Templates are a simple way of storing a calculation method for a Metric. They are a full business logic component and are a handy way of creating a baseline for other business logic components. New business logic components created from a template copy the code and create a new instance of it. However, in general, flexibility when using templates is quite low, and Business Logic Modules should be used wherever possible.

Business Logic modules are independent code components that enable re-use of the same code base by other Business Logic. Modules can also include other Modules, so the hierarchy levels can be multiple. When using Modules, the code is contained in one place and is re-used by each of the other components which link to it. This re-use of code sections eases the maintenance by eliminating code duplication and making it possible to apply system-wide logic changes quickly and easily.

During the design stage, it is necessary to identify the main Business Logic modules and their related parameters. Once the contract modeling is completed and the Contract Manager has a clear view of the logic to be used, it becomes possible to identify the calculations they have in common and that can be defined in separate Modules.



The above diagram depicts a Module that calculates the success rate of helpdesk activity to meet a target within given thresholds. To implement it as described, it is necessary to define two parameters, known as Metric Parameters: one that defines the type of helpdesk activity, and another for the threshold to compare against (see definition of *Metric Parameter* in Cases to be Considered During the Modeling Process (see page 32)).

By careful consideration of the types of calculations implemented in the system, you will probably find that a number of similar types may be performed by changing one small section of code, and using a parameter to act as the 'switch' between them. In this way, you can minimize the amount of code you need to create, and maximize the amount of code re-use.

Service Level Templates

A Service Level Template is a packaged collection of service components and the associated Metrics defined to measure them. These service level templates can be created as required and are often defined according to the most commonly measured service aspects.

The key issue when defining service level templates is that all possible parameters that could be used to change the behavior of the Metrics should be identified and exposed. This provides the greatest flexibility for the system and facilitates configuration for the users of the system. When using the service level template to create new Metrics, each of the configuration parameters is exposed to the Wizard interface, meaning that little or no further customization is required in order to activate the Contract. The parameters exposed to the user using the wizard are in the objective statement. Therefore, consider in the objective statement of the Metric all the parameters you want to be available for changing by the end user.

To provide for maximum efficiency with service level templates, you should aim to perform all of the Business Logic through the Module functionality as described previously.

Following is an example scenario for using service level templates, where Application Hosting service components are provided to customers at the following three levels, Bronze, Silver, and Gold, depending on the amount that the customer wants to pay. Additional Metrics can be provided in each higher hosting level, along with more stringent targets. Each of these levels could be a good candidate for creating a service level template, as shown in the following sample scenario:

- Application Hosting - Bronze (5 Metrics)
- Application Hosting - Silver (8 Metrics)
- Application Hosting - Gold (12 Metrics)

Now, when a new customer is added to the system, it is easy to choose the required definition based upon what the customer wants. Using the wizard interface enables each of the Metrics within it to be customized for that particular customer. Note that it is also possible to choose only some of the Metrics from the definitions, or even add Metrics from two different definitions to the new customer's Contract, if some special customization is required.

Importance of a Strong Service Catalog

As described previously, the service catalog is a key component of the system and should be configured in a structured manner. This enables the system to be used effectively by all users and avoid confusion. It also enables the system to grow with the organization and handle future enhancements and additions with minimal impact on what has already been created.

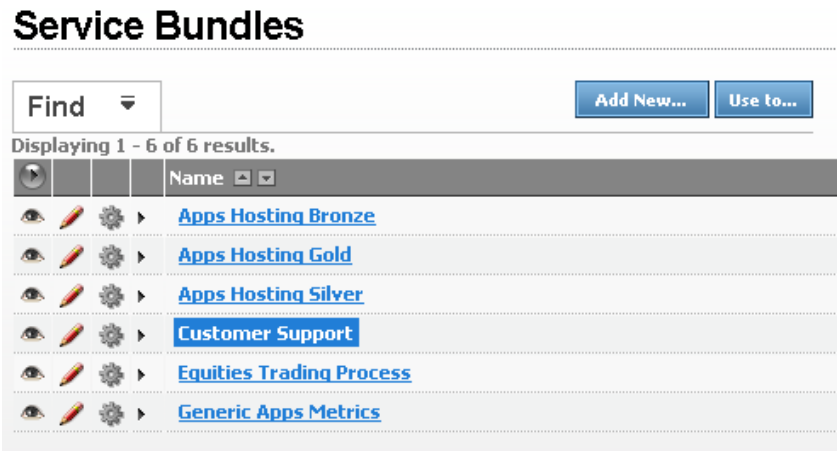
The system offers a high degree of flexibility in creating and managing the catalog of service components and SLAs. However, since it can only be as good as the design, spending the time to plan for the future is essential at the early stages of design.

Defining the CA Business Service Insight service catalog in an efficient and structured way provides your system with the flexibility to add services and domains to the service catalog later on. It also enables adding Contracts, Metrics, alerts, and reports in the future. In addition, it leads to a more structured approach to the business logic and paves the way towards being able to use standardized modules and templates to handle business data and associated calculations.

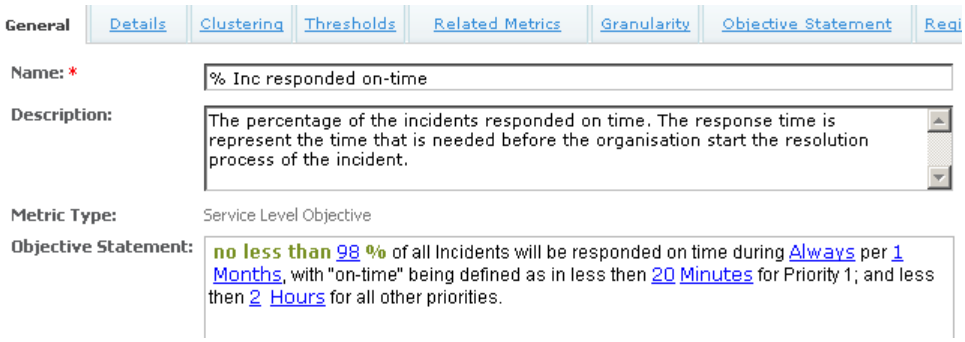
In conjunction with the catalog, the service level templates defined in the system provide an excellent way for the Contract Manager to create new Contracts very easily and with little or no knowledge of the underlying levels of data required. After it has been set up effectively, it should be possible to configure a new Contract for a customer by modifying only the parameters in the service level template. This all relies, however, upon the catalog and definitions being set up most effectively. These parameters are all exposed via the objective statement for each Metric in the service level template and can be modified either there, or from the wizard when using the definition.

***Example*:**

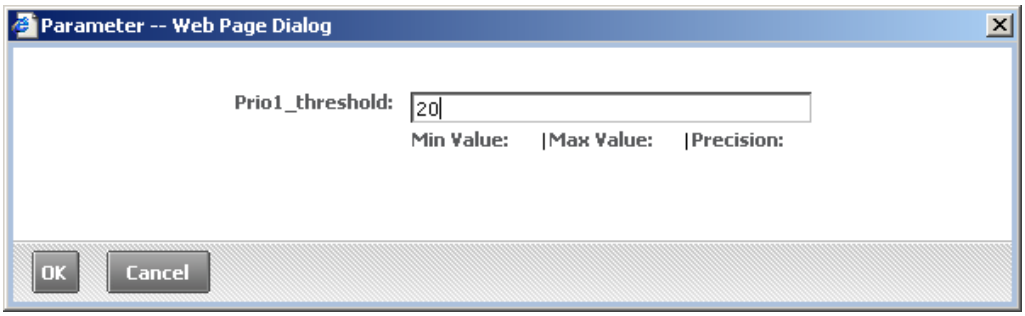
When using a service level template to deploy some Customer Support Metrics to a Contract, it is possible to select required metrics from an existing definition.



Inside this service level template, there is a Metric called '*% of Incidents responded to on time*'. You can see that there is a level of subjectivity here, in that the '*on time*' component may be open to question. The following example explains the measurement made in this Metric:



The objective statement displayed at the bottom of the Metric's General tab (or alternatively on the Objective Statement tab) shows all of the parameters exposed in this Metric. In the previous figure, the definition of '*on-time*' is given as 20 minutes. This is a parameter which is customizable to allow our own reinterpretation of this pre-defined Metric. To change this value, you can click the 20 Minutes parameter link.



In this way, the new Metric created from the service level template can be customized without the need for modifying the underlying business logic of the Metric. Note that this also assumes the business logic is written in such a way as to incorporate these parameters in the service level calculation.

This simple example shows clearly how important it is to create a powerful and flexible set of service level templates for the system catalog to enable leveraging future Contracts from them.

Financial Management (Penalties, Incentives and Costs)

In earlier versions of CA Business Service Insight, there were contract entities known as Penalties which were implemented using Excel-like formulas. Penalties based their results solely on the input from the contract's Metrics and relied on basic functions to calculate the resulting penalty amount. From version 4.0 and up, these have been replaced by full Financial Metrics which can be created by the user, allowing greater flexibility. These financial Metrics can be used to provide Incentive or Cost information about the contract.

Note: An Incentive replaces the old Penalty terminology from CA Business Service Insight 3.0 and earlier, and can be positive or negative depending on performance. A negative Incentive however, is basically just the same as a penalty. It is also important to note that if you are implementing a Penalty with the Incentive Metric Type, you *must* remember to make the Result() function return a negative value. This enables any summarizing functions which may combine different Metrics results together, to adjust the totals in the right direction. Tha is, an incentive increases the value whereas a penalty decreases it.

CA Business Service Insight version 4.0 also provides the ability to create a Consumption Metric measuring the usage of service components and resources, and also to combine this with a Price Item Metric to determine the cost of that service or resource. Combining these with the enhanced forecasting functionality makes it now possible to create some very comprehensive Financial Management Metrics.

Financial Metrics can also acquire the output from other Contractual Metrics and determine the related Penalty or Incentive values based upon the performance of these contractual Metrics. They can also work with other types of information to determine their result, such as Price per Unit information and forecasting models enabling reporting functionality such as Expected vs. Actual Costs.

***Cost Item Example*:**

A particular Risk Application has an associated cost based on the number of concurrent users of the system. It is calculated monthly and there is a forecasted value provided for this application. The price per unit of this application (cost per concurrent user) is provided in the table below (assume this application falls under index 1):

Index	Oct-01-2006	Nov-01-2006	Dec-01-2006	Jan-01-2007	Feb-01-2007	Mar-01-2007	Apr-01-2007	May-01-2007	Jun-01-2007
1	1.0	1.0	1.0	0.75	0.75	0.75	0.6	0.6	0.6
2	0.9	0.9	0.9	0.7	0.7	0.7	0.5	0.5	0.5

The forecasted number of concurrent users for this period is also available (Again, index 1).

Index	Oct-01-2006	Nov-01-2006	Dec-01-2006	Jan-01-2007	Feb-01-2007	Mar-01-2007	Apr-01-2007	May-01-2007	Jun-01-2007
1	3000	5000	8000	5000	5000	5000	5000	5000	5000
2	2500	2500	6000	4000	4000	4000	4000	4000	4000
3	1000	2000	3000	1000	1000	1000	1000	1000	1000

By modeling this Cost Metric using these costing tables it is possible to determine the monthly application cost, based upon the actual number of concurrent users. This information comes from the data-source and can be multiplied by the price-per-unit figures above to give the Cost figures. It can also be compared to the forecasted values to provide an analysis of the expected vs. actual cost. The Business Logic in this case is required to determine the actual number of concurrent users encountered during the period and multiply it by the cost per unit values. In addition, the forecast function within the Business Logic refers to the Forecast information. This is an example of applying a Cost Item metric.

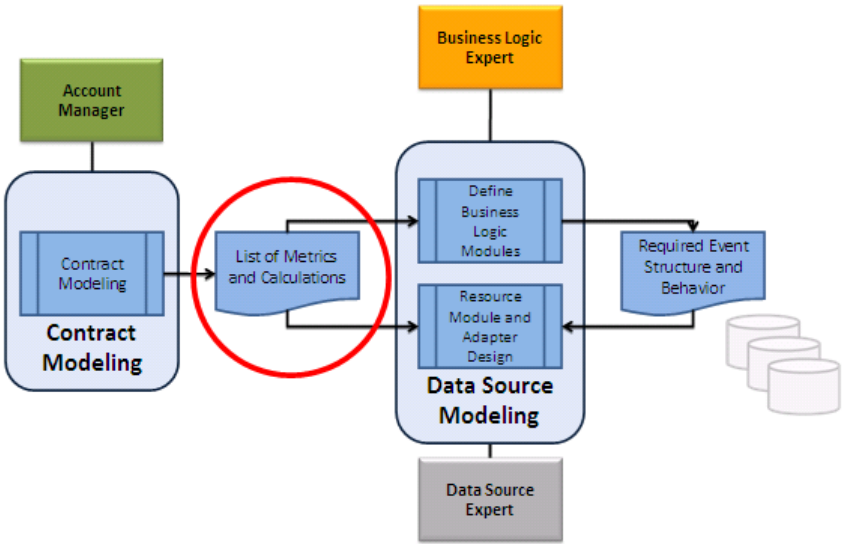
***Penalty scenario Example*:**

The customer SLA has a nonperformance clause included in it to ensure that the network is available 98% of the time for any given month during business hours. Any monthly service level below this incurs a penalty payment based upon a formula (Penalty = $1000 per each whole percentage below the target (i.e. 96.5% = (98-Round(96.5)) * 1000 = (98-97) * 1000 = -$1000)).

In order to implement this penalty condition, you can create a Financial Incentive metric which takes its input from an existing metric (*Network Availability* >= 98%) The registration process for this Metric uses the RegisterByMetric() process to receive the service level values from that Metric in order to perform the comparisons. This sends the service level values for the tracking period of that Metric into this financial metric as events which are then used as part of the calculation to determine the Penalty amount for that same period using the formula from the scenario.

Note: For further case studies, refer to the Financial Metric Modeling Case Study (see page 224).

Contract Modeling Stage Outputs (Contract Manager, Data Source Expert)



As shown in the previous diagram, in order to proceed to the Data Modeling stage of the solution, the Contract Manager is required at the end of the Contract Modeling stage to provide the List of Metrics and their required calculations. The list includes the following information:

- A completed Service catalog definition, comprising:
  - Complete list of service components to be offered
  - Breakdown of the service domains and domain categories
  - Definition of all required timeslots for each of the Metrics
  - List of Business Logic modules to handle each type of calculation (including the parameters needed to implement them)
- A completed list of Metrics to be implemented, including:
  - Well-defined naming standards for Metrics
  - Categorization of each metric according to pre-defined service catalog components

This document is a useful tool for ensuring that all of the key definitions for a Contract have been made and all of its Metrics are fully defined.

The information within this spreadsheet is the input that the Data Source Expert needs in order to proceed o the next stage, Data Modeling.

After these items are completed, it is possible to proceed to the next stage, where you can begin modeling with actual data from the sources. Without a completed Contract Model, it is very difficult to know exactly what is required from the data source in order to satisfy the contractual obligations.

Data Modeling (Data Source Expert, Business Logic Expert)

The Data Modeling section describes the second part of the design process. The data modeling phase is the process of taking data from customer sources, identifying specific components of the required data, deciding how this data needs to be normalized, and assessing how to assign the relevant data to the corresponding Metrics (via the registration process).

This phase includes the following tasks:

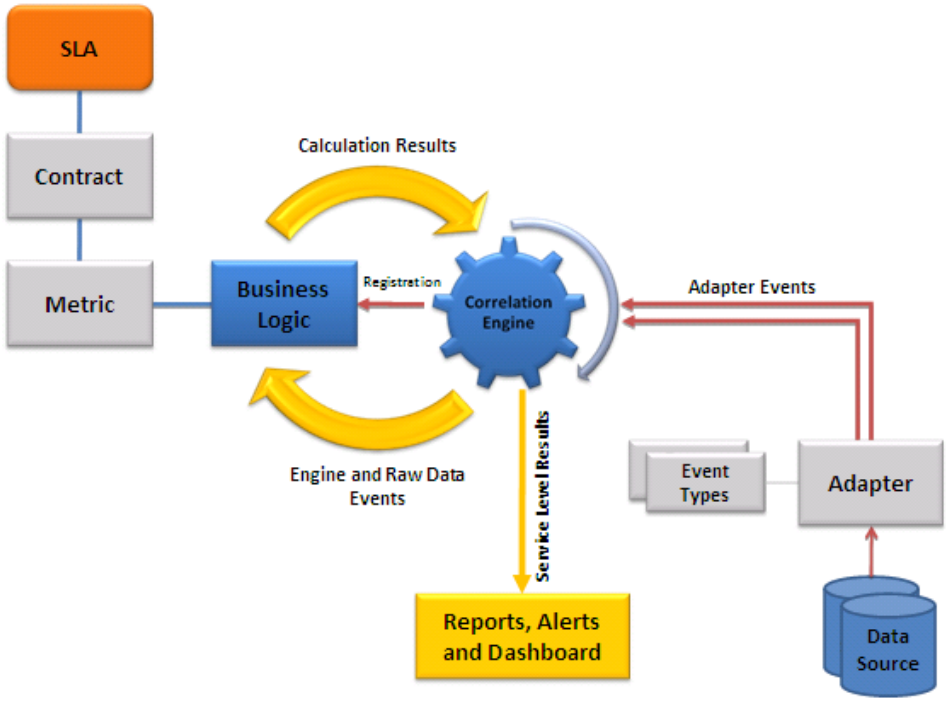
- **Business Logic Expert**:
  - Identifying and defining input event structure required for calculation later to be defined as Event Types
  - Building required fields into Event Type to provide data required for all calculations in which it is used
  - Defining Metric registration process to maximize events flow
  - Determining how to build the resource model from available data to satisfy all reporting and logic requirements
- **Data Source Expert**:
  - Identifying data source type and deciding how it should be normalized by the Adapter into defined Event Types and fed into CA Business Service Insight database
  - Deciding Event type identifiers that should be attached to data

Events and their Flow

The data flow within the system is in the form of Events. Where an Event is, an information message created by the Adapter based on source data and is in a format that can be used by CA Business Service Insight for its service level calculations. Raw data always consists of Events.

The focus of the design must therefore be on this Event flow within the system.

Before modeling the data requirements, both the Business Logic Expert and the Data Source Expert need to have a solid understanding of Events and their flow within the CA Business Service Insight system. The following diagram illustrates in high level this basic Event flow.



The previous diagram depicts how Events are retrieved from the data source by the Adapters and normalized into a standard Event structure defined as the Event Type. These Events are sent by the Adapters to CA Business Service Insight. These Events are referred as Raw Data Events.

The Business Logic calculations in each Metric are based on a subset of the Raw Data Events. The Business Logic therefore requests this subset by performing *registration*.

Based on the registration statement, the Correlation Engine sends only the Raw Data Events relevant for business logic calculations.

Additional types of Events sent to the business logic are the Engine Events. All concepts involved in this process are discussed in detail in this chapter.

This section focuses on the following parts of the diagram:

- Data source
- Adapter
- Event Type(s)
- Metric registration

The CA Business Service Insight Data Model has been designed to maximize the efficiency of this stream of data within the system.

In general, CA Business Service Insight functions on two layers: The Infrastructure layer and the Business Model layer. As a simplistic breakdown, the infrastructure layer includes Adapters, Resources, and Event type objects, while the business layer includes Contracts, Metrics, and Services objects. Between the two layers there is a virtual shim layer, called the Correlation layer.

One Event identifier is the Event Type object. The Event type determines how Events are defined and how they are reported to CA Business Service Insight. It also defines the structure of the Event data field so that it can be interpreted by the Business Logic during processing.

Another Event identifier is the Resource which is the smallest entity used in calculations. For example, when calculating server availability, the logical definition of the smallest entity on which reporting is required might be a specific server, or it might be a customer when reporting on that customer's ticket handling. The Resource is a definition of an CA Business Service Insight entity that is derived from both the data source and the calculation requirement. Each Resource is given a Resource Type which is Resource identifier that determines exactly what 'type' of resource is defined. Each Resource must have a Resource Type associated, which also allows the addition of custom attributes to be associated with each resource. For more information about these attributes, see Resources and their Management (see page 51).

The correlation occurs between incoming Adapter Events and Contract Metrics. The heart of this correlation process is the Resource allocation and Metrics registration.

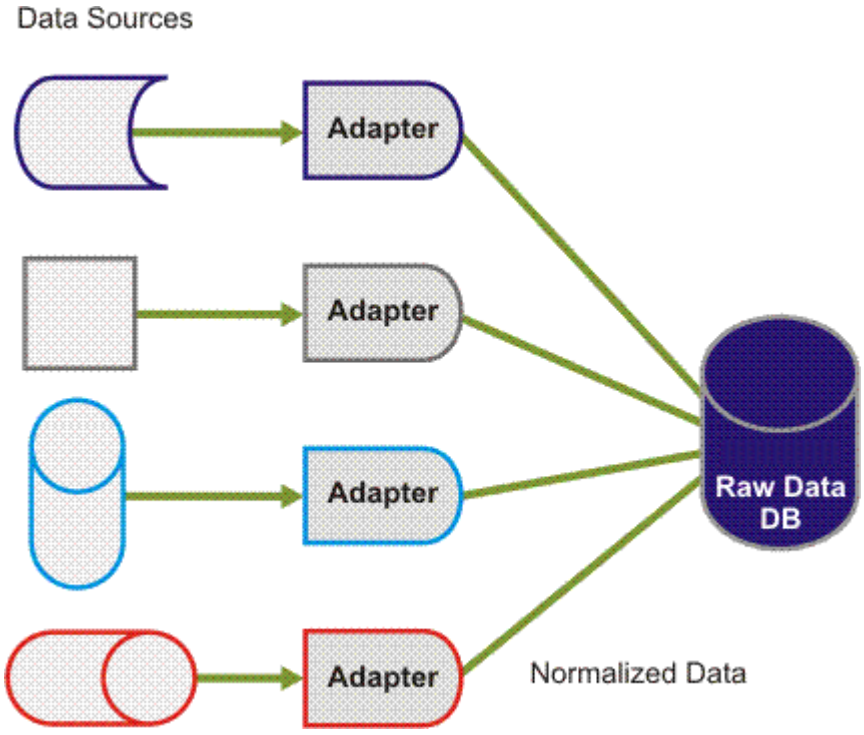
Resource allocation and Metrics registration specify which resource event streams are measured and by which Metric.

Note here that with Metric registration, there may be a degree of re-use and co-dependency with other Metrics since it is possible to use the output of one Metric as the input to another one. Similarly, there are Interim events which are not used as the output of a Metric for service level measurement, but more as an intermediate calculation step which can then be used by other Metrics.

Data Model - Overview

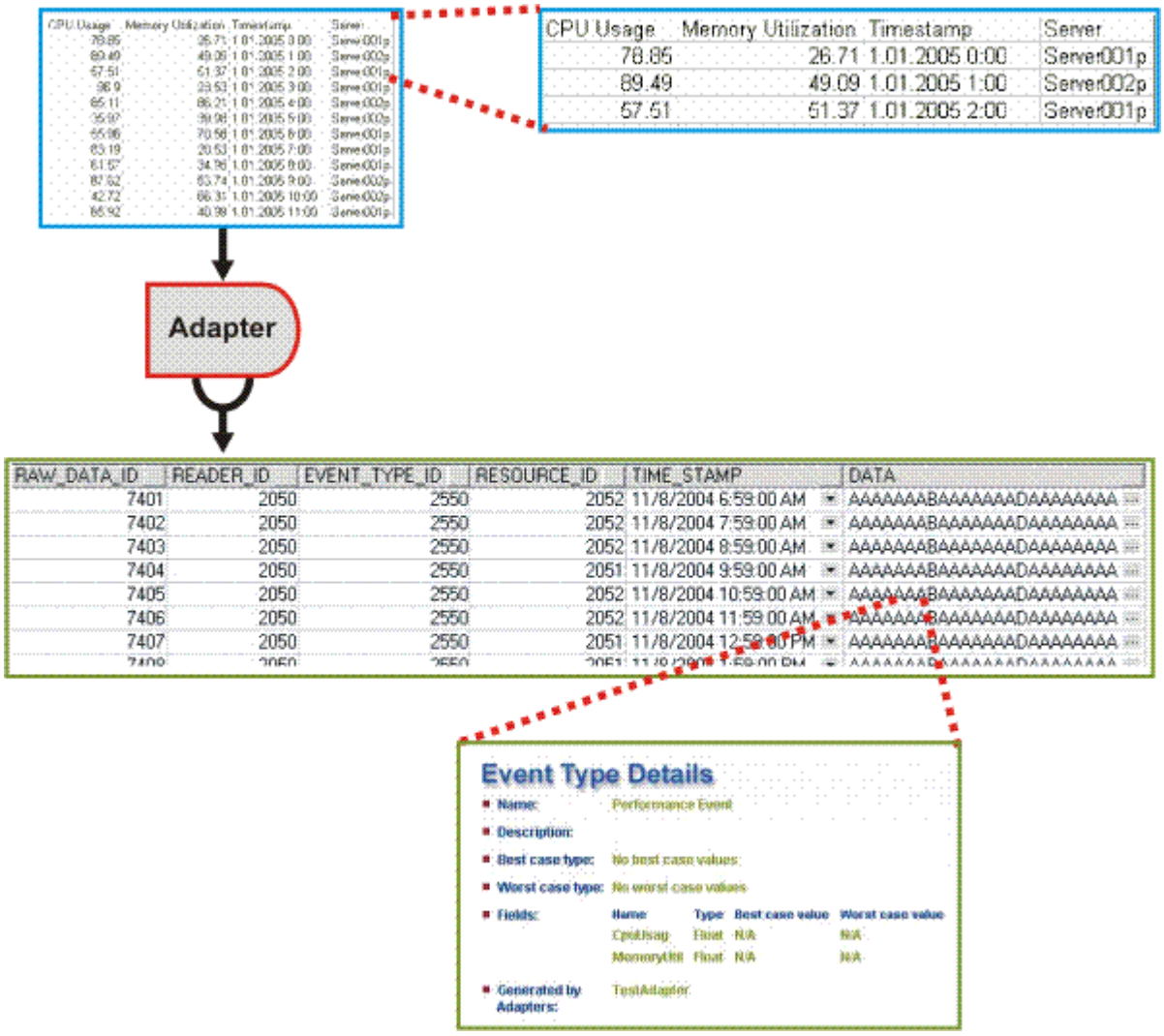
The CA Business Service Insight data model has been designed to meet and overcome the following challenge.

Raw data is retrieved by the Adapters from various, disparate data sources and held in a variety of formats. This diverse data needs to be retrieved and homogenized into a single database table. Therefore, the Adapters are required to read and normalize the data into a unified data model, as shown in the following figure.



As part of this process, all data fields are inserted into the same database table field, but they are encrypted. Each line inserted into the CA Business Service Insight database has an Event Type identifier attached to it. The Event Type definition contains descriptions of the data fields. It also enables the Correlation engine to correctly interpret the data fields and identify when they are required by the Business Logic for calculations.

The following figure shows a graphical representation of the data retrieval and database population section of this process. Also depicted is an enlarged section showing what the data represents in real-life terms, rather than how the raw data appears.



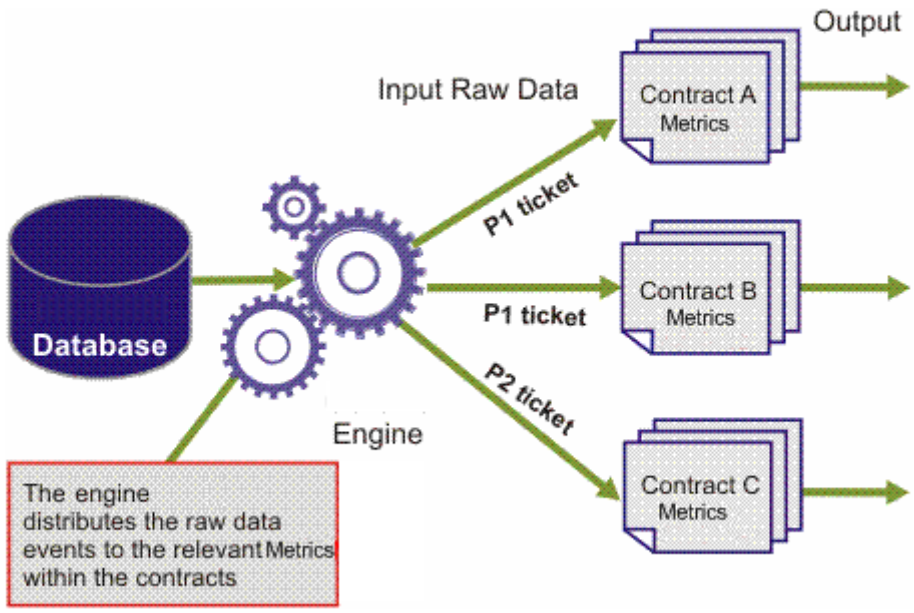
The CA Business Service Insight system also includes all Contracts and Metrics that require evaluation against the raw data to produce resulting service level performance information. Each Metric must receive only the subset of data relevant to its calculation. The raw data holds a potentially vast number of records of various types. Using the Metric to filter the relevant Events by their values is very inefficient. Therefore, the CA Business Service Insight engine distributes the relevant raw data to each specific Metric.

***Example*:**

For the following two Metrics in a Contract:

- Priority 1 (P1) tickets average resolution time
- Priority 2 (P2) tickets average resolution time

The first Metric is required to evaluate only those tickets with Priority 1, and the second Metric, only Priority 2 tickets. Therefore, the engine needs to distribute the records accordingly. In addition, the resolution time within one Contract is calculated for P1 tickets that are opened for contract party A, while in the second Contract, P1 tickets for contract party B, and in the third P2 tickets for contract party C. Therefore, the engine is required to select the ticket type and customer for whom it was reported, as shown in the following figure.



As previously explained, the raw data records have attached identifiers that allow the engine to identify the records and Events relevant to each Metric's Business Logic. The two identifiers are the Event type and the Resource.

Adapter Translation and Normalization

The function of the Adapter is to read the data from the data source and normalize it into the format of an Event. Each Event in CA Business Service Insight is composed of the following fields:

- Resource ID
- Event Type ID
- Timestamp
- Value Field/s according to Event Type

The Adapter needs to link the original fields in the data source into the corresponding CA Business Service Insight Resource fields. To do this, it uses a translation table that contains the value found in the data source and corresponding CA Business Service Insight Resource ID.

The process of attaching the Resource ID and Event Type ID to the relevant data source value is called *Adapter translation*. During this process, the translation table is created with the matching values. This table is used by the Adapter in order to populate the relevant Event Type ID and Resource IDs in the Event record that it is creating. Independent tables are created for translating Resources and Event types.

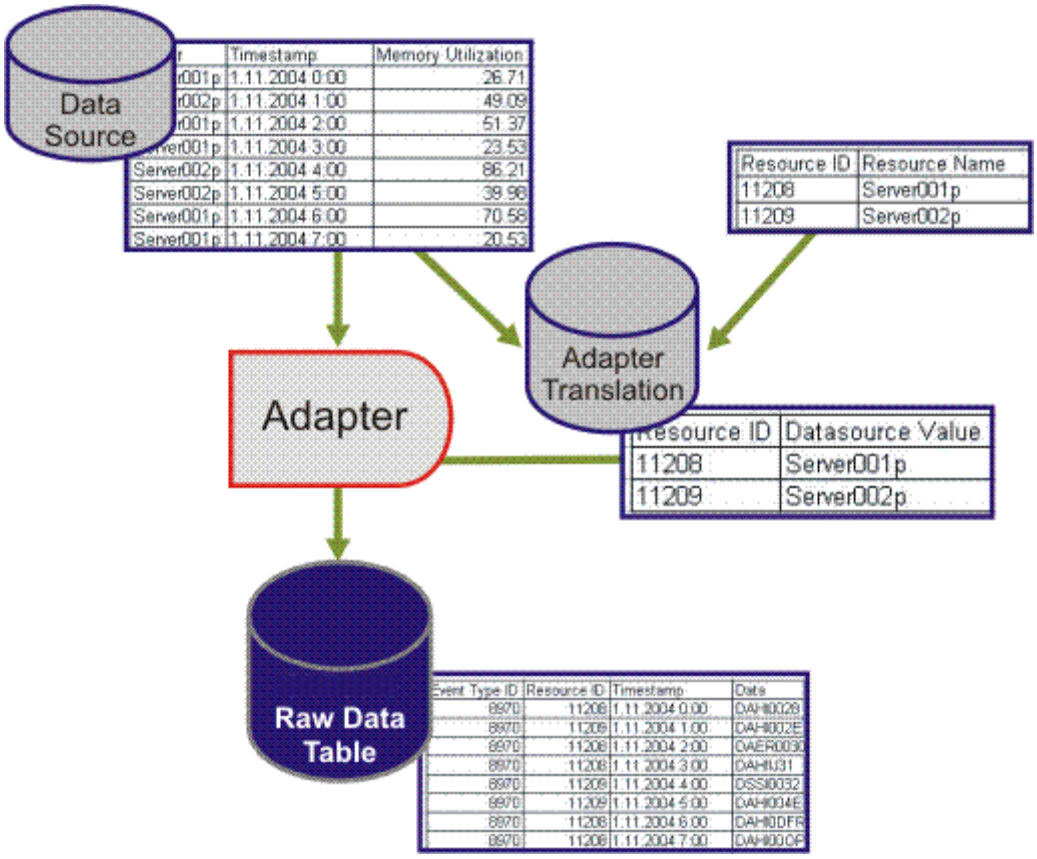
As mentioned previously, the Resource ID and Event Type ID are used for registration purposes, while the data field and Timestamp values are used for actual calculations.

The Timestamp field is also used by the engine to determine the order and timing of Events sent for calculations.

Event Type definition is done manually in CA Business Service Insight based on the input data source and required output.

Note: Resource definition can be performed either manually or automatically using translation scripts (see Resources and their Management (see page 51) for additional details).

The following figure shows the interaction between the data source, Adapter translation table, Adapter, and CA Business Service Insight raw data table.


Data Source
Timestamp
Memory Utilization
1.11.2004 0:00
26.71
1.11.2004 1:00
49.09
1.11.2004 2:00
51.37
1.11.2004 3:00
23.53
Server002p
1.11.2004 4:00
86.21
Server002p
1.11.2004 5:00
39.98
Server001p
1.11.2004 6:00
70.58
Server001p
1.11.2004 7:00
20.53
Resource ID
Resource Name
11208
Server001p
11209
Server002p
Adapter Translation
Adapter
Resource ID
Datasource Value
11208
Server001p
11209
Server002p
Raw Data Table
Event Type ID
Resource ID
Timestamp
Data
8970
11208
1.11.2004 0:00
8970
11209
1.11.2004 1:00
8970
11208
1.11.2004 2:00
8970
11208
1.11.2004 3:00
8970
11209
1.11.2004 4:00
8970
11209
1.11.2004 5:00
8970
11208
1.11.2004 6:00
8970
11208
1.11.2004 7:00

Metric Registration

In order for the Correlation engine to know what data may be requested, the Metric must register its presence and requirements with the Correlation engine.

Metric registration is the request by a Metric to receive Events, and only those that need to be included in its calculation. This request is done by the state of the Event identifier's Event type and Resource.

Registration can be done on the basis of a single Resource or a group of Resources.

***Example*:**

For the following informational metric: "Number of down-times of Server X" and on the assumption that the data source provides a notification when a server goes up or down, and that the notification indicates whether it is up or down at a particular time, then the Registration is:

Event Type: Server Up/Down Event

Resource: Server X

Based on the above registration, what happens is that the engine filters out all Events that have up/down definitions in their Event type field in addition to having Server X in the Resource field.

Once a Contract is activated in the CA Business Service Insight system, all the Metrics register the relevant Events necessary for their calculations. Based on these requests, the correlation engine marks the Events relevant to each Business Logic. Once calculations begin, the relevant Events are sent to each Metric for calculation.

Resources and their Management

In order to allow Registration to be dynamic, Resources can be allocated either individually by their own unique name/identifier or by their relation to logical groups.

***Example*:**

For the Metric: "Overall number of down-times of data center servers"**,** the registration is:

Event type: up/down Events

Resource: All resources that are tagged under servers of data center.(This would probably be a resource group.)

Understanding the Resource Life Cycle

A Resource is a physical or logical entity that can change its characteristics over time. It may be allocated to certain service components or contract parties, and so on, at one time, and then be reallocated at some point in the future. Each of these changes or re-allocations are captured by CA Business Service Insight in order to be able to perform calculations at any point in time, based upon resources configuration and allocation at that exact time.

Changes to a Resource or its allocations can be made at any time, but require that a new version of that resource is created. Each new version needs to have an effective date set for it, for when the changes will occur. The changes will then carry forward into the future, unless additional changes are encountered in a later version of that same resource. All changes will only be visible and available for the calculation engine once this new version has been activated and is in effect. This process is called 'Committing' the resource.

Within CA Business Service Insight, there is also a way of handling multiple Resource allocations in one step. This method is via the use of a 'Change Set'. Change Sets allow large volumes of resource changes to be made in a single 'transaction', similar to the way a transactional database works. All the changes can be made to all resources which are allocated to a Change Set by performing the operation on the Change Set as a whole, and then committing the change set in one step.

When dealing with resources and their changes, it is worth considering the following points in relation to the calculation engine:

- When activating (committing) changes on a specific resource or on a group of resources (change set) try to consider what will be affected in the system. Since changing the resource module might trigger a recalculation, it is important to optimize the resource(s) activation date and the number of changes activated in one operation.
- **Mass update**:- It is possible to apply the same change to many resources (mass update). Since changing the resource module might cause recalculation, it is important to optimize this operation.

The previous example addressed a Resource not directly, but by its logical allocation to its function or location (in this case, to its function, a data center server).

The Registration request could be very cumbersome if Events are requested for each individual server held in the data center. One problem is the number of Resources to which are referred. The other is that the infrastructure of the data center changes on a regular basis, so that a server that was a part of the data center may no longer be there, or a new server may be added. Therefore the list needs to be dynamic.

Based on the previous example, it is clear that the Resources need to be attached to a logical group so that they can be addressed via this logical entity. Further, the logical group itself may need management if it is constantly changing.

Resource allocation is the CA Business Service Insight method of tagging Resources. A Resource can be allocated to one or more groups, Resource Types, Contract Parties, or Services. Resource allocations are managed using CA Business Service Insight Version Control.

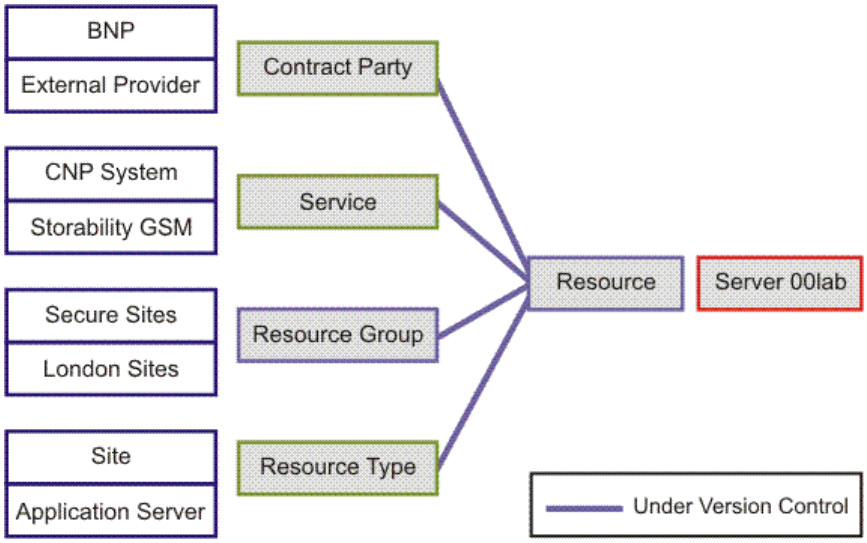
The Resources available for inclusion in calculations are determined by the resources currently in effect within the system (in relation to the time range being calculated at the time).

Now, returning to the previous example:

Overall number of down-times on data center servers

The data center can be represented in the system as a Service to which is then allocated all servers within the data center. It may also be defined as a Resource Group called "data center servers". These are two alternative methods to choose for the resource allocation in this particular case, but there are more options available.

The following diagram demonstrates to which Entities a Resource may be attached and their logical uses.



A Resource group can reflect any aspect of the Resource that is required for calculations, such as its location or the technology it contains.

The main purpose of allocating Resources to these entities is to ensure both the correspondence with the calculation requirements and that the model remains as dynamic as possible.

Custom Attributes

Another feature available for a Resource is the ability to provide custom attributes. Each Resource Type can have these custom attributes added to them, and in turn, each Resource defined as being of that particular Resource Type, inherits these custom attributes.

Using the previous example, for each of the data centre servers, associate also their IP Address. If each of the data centre servers is defined with the Resource Type of 'Data Centre Server' then ensure that a custom attribute called 'IP_Address' is added to the Data Centre Server Resource Type. In this way, each Resource (server) can then be associated with its IP address via the custom attribute.

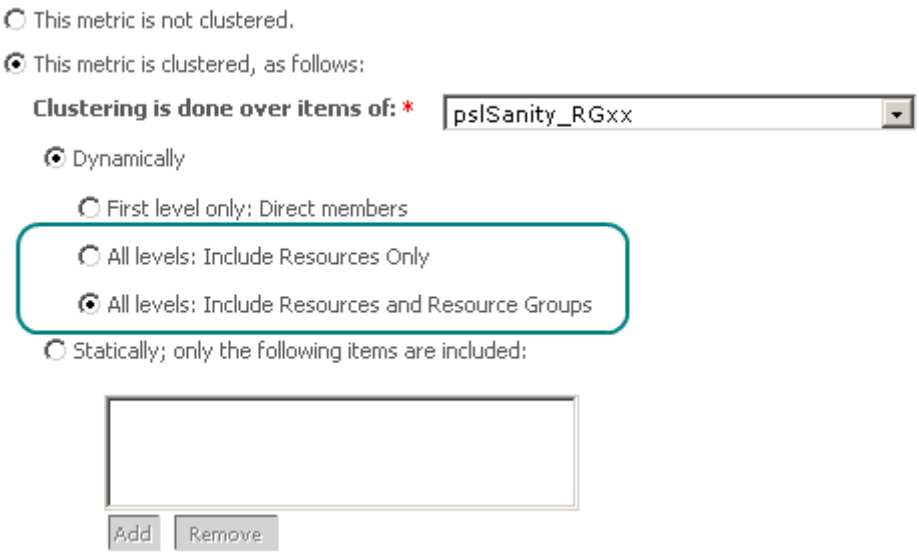
Note: For more details and a sample case study refer to Case Study Using Custom Attributes (see page 241).

What is a Clustered Metric?

A Clustered Metric is a Metric set to calculate service levels for a group of Resources. The calculations are done for each of the Resources individually, without the necessity to duplicate the Metric every time with the registration of a different resource.

The clustering can only be done on a group of Resources that either have no target service level attached to it or have the same target service level. For example, the availability of all application servers should be at least 99.9% per server. In this case, a Metric is clustered over a group of resources that contains all application servers. The Engine calculates a separate service level result for each server, where the target for each server is 99.9%.

In addition to this type of clustering, CA Business Service Insight supports a 'roll-up' type of clustering, enabling multiple levels of resource (and group) reporting from a single metric. This enables the service level to be calculated at multiple levels of the resource hierarchy and a drill up/down feature between this resource structure's related entities. From the Metrics Clustering tab, you can enable this option by selecting the relevant options on this page.



Building the Data Model of the System

As part of the Data Modeling process, required components are identified based on the data source and calculation requirements.

Following is a list of the components that should be identified in the Data Modeling process and their definitions.

Event name Name of Event as it appears in CA Business Service Insight; should be as descriptive as possible.

Event Behavior Behavior of specified Event; when received from the data source, what conditions, and so on.

Timestamp field Field in data source used as Event Timestamp.

Event type field Field in data source to be translated into an Event type, describing type of reporting. Important that number of different Event Types are minimized as far as possible because Event Type definition is manual and should ideally only be done once.

Data fields Fields in data source to be retrieved as data fields.

Resource field Field in data source to be translated into a Resource. Contains an entity required to report on with a relatively fixed life. Resource is an entity with a defined life-cycle, where changes can be managed dynamically within the system. Referring to a resource life cycle indicates how frequently new resources are added and changed in allocation to different Service or any other entity of allocation, as mentioned above. The field to be translated as a resource in CA Business Service Insight should have few allocation changes and limited other ones.

And lastly, based on all of the above definitions:

Registration by Defines registration criterion. Registration criterion defines Event Type and Resource the Metric registers. Request for Resources can be done directly by registering by Resource, or by allocation entity, such as Service, Contract party, Resource group, Resource type, and so on. This definition is determined by registration capabilities.

Another method is Register by Metric, where the current Metric obtains outputs (service level result) from another Metric and use them as input. Possible also to use result from multiple Metrics as input.

Guidelines for Defining Registrations

Use the following guidelines for defining registrations:

- Never set the registration to be only by Event Type. Even if the calculation requirement does not require Resource filtering, add the Resource filtering at least on the Resource type.

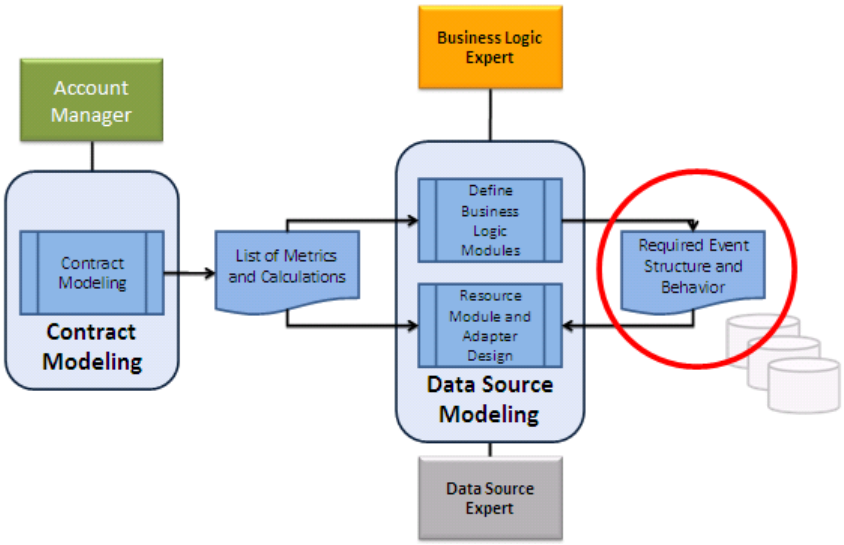
  For example, where calculating overall average response times of application servers, the Response Time Event needs to be reported only when it is associated with one of these application servers (that is, where the Resource Type of the Resource associated with the event is 'Application Server'). In such a case, the system may have other types of Resources such as sites or routers using the same Event Type to send data, so to distinguish between them. The Resource Type to be reported on ('Application Server') should be added to the registration command as the 3rd parameter.

  The reason for this is that when a Resource change occurs, the Engine marks the Metrics associated with this Resource as 'requiring a recalculation'. In a case where the Metric registers only by Event type, the Engine views the Metric as registered to all Resources and therefore marks it for recalculation upon activation of each Resource. To avoid this, you must use the 'optional' 3rd parameter of the resource type.

- The most efficient method for registration is by Contract Party and Service. Arranging the Resources in this manner enables expressing the logical relationship between the data layer and the business layer in the system. Registering resources through these entities does not require changes to the formulas when used in different Contracts or when used for different Services. The Metric context contract and service defines the relevant contract party and service. The Business Logic formulas defined in this type of registration are easily reusable because they do not require any changes in the registration.

Note: Registrations can also be handled by the *Registration* tab within each Metric. This interface provides a wizard which guides you through the process for the selected metric.

Data Modeling Stage Outputs (Data Source Expert and Business Logic Expert)



In order to proceed to the Implementation stage of the solution, the Data Source Expert is required to have the following:

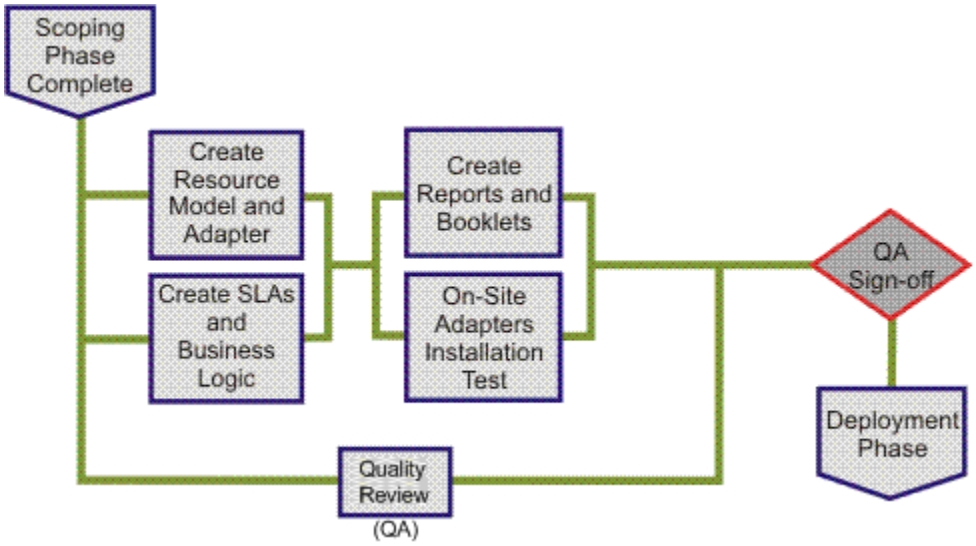
- A thorough understanding of the data sources to be interfaced with, including:
  - Sample historical data
  - Communication method
  - Understanding of key fields from each data source to be used for key components of the resource model
- For each data source is a inherent list of Event Types describing structure and behavior, including the following definitions:
  - Event name
  - Event Behavior
  - Timestamp field
  - Event type field
  - Data fields
  - Resource field

Note: You should create a summary document of the information you acquire.

Chapter 3: Implementation

This section contains the following topics:

Implementation - Introduction



This chapter explains the process and the reasoning behind the implementation phase of the project. As shown in the previous figure, the Implementation phase follows the Design phase, which is followed by the Installation and Deployment phase.

The objective of the implementation phase is for the Contract Manager to be able to complete the actual creation of all items and objects in the CA Business Service Insight system that were previously defined during the Design stage. During the implementation phase, the team is involved in preparing the system for full deployment and installation.

This phase should not be started before the full Design phase is complete and all necessary objectives have been taken care of and correctly defined. If the Design phase has not been completed correctly or not all Contracts, Metrics, Adapters, and so on have been clearly defined, there will be problems with the system or items missing that should have been implemented during the implementation phase. The implementation phase should only start after the Design review sign-off.

It is also important that the implementation phase is completed correctly before proceeding to the installation and deployment of the system. This should only be done after performing a quality review.

The configuration process involves the Contract Manager performing the following steps:

- Service Catalog setup
- Creating Contracts
- Activating the Contracts
- Configuring Security settings
- Creating Reports/Alerts/Dashboards

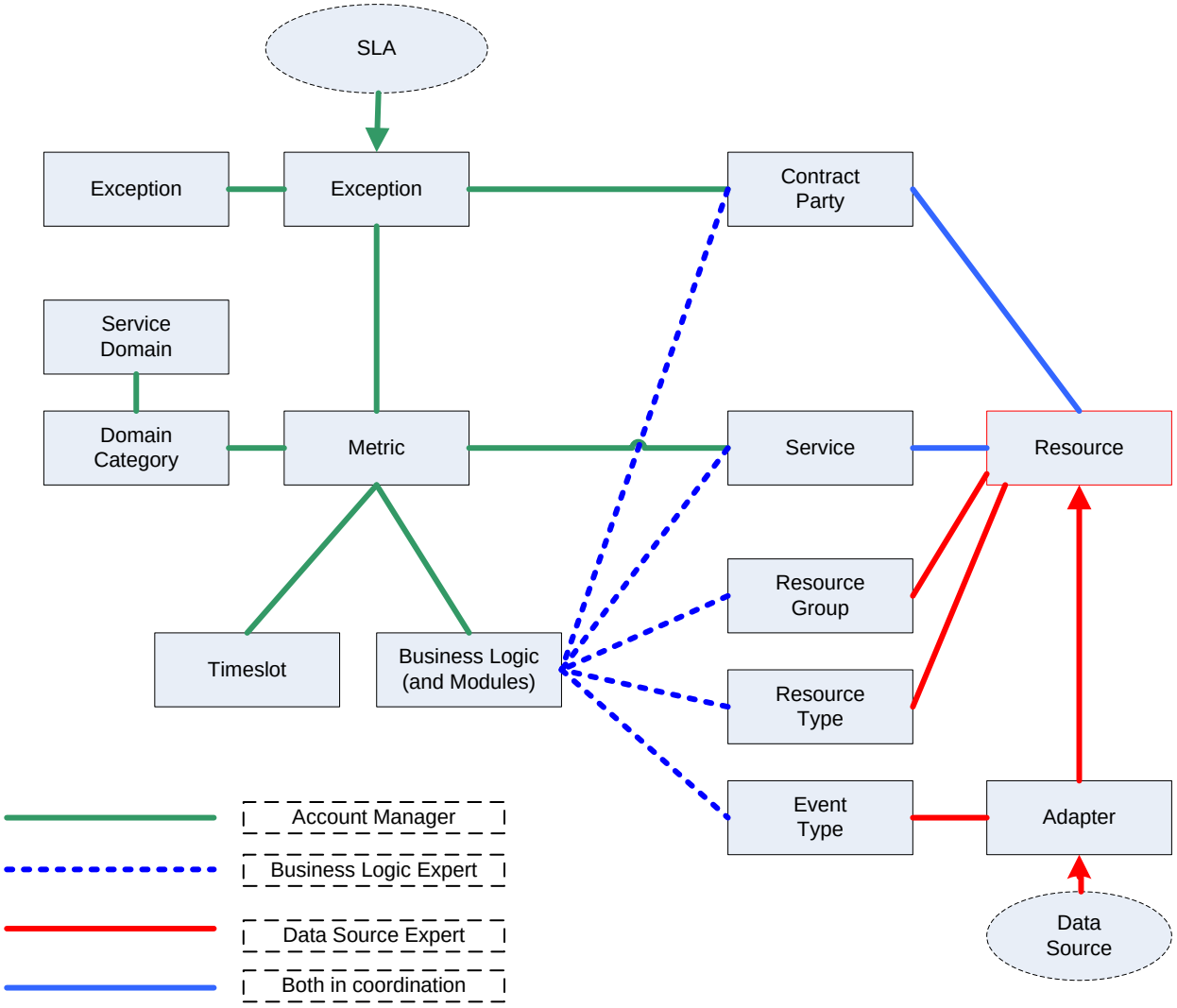
The Data Source Expert should perform the Adapter configuration and integration with the data sources. The Data Source Expert must also perform the resource translation step to link the data source structures to the entities defined within the CA system. This step is crucial to the whole process and may require some guidance from the Contract Manager also.

In addition to this, the Business Logic Expert is required to write the Business Logic for all of the Metrics, according to the plans from the Design phase. This may include creating all required modules and configuring associated parameters to provide the expected calculation functionalities.

All of the above points are explained in further depth in the sections within this chapter.

Note: It is important for the Contract Manager to be aware that poor choices made at this stage can adversely impact the operation of CA Business Service Insight and may be difficult or impossible to reverse at a later stage.

The following figure outlines the overall logical work-flow.



Setting Up the Framework (Contract Manager)

The Framework allows:

- Defining Services, Service Groups, Service Domains, and Units
- Creating and maintaining templates, including Business Logic and Timeslot templates, and Business Logic Modules
- Managing Custom Attributes for the Resource Types

At this stage, all system-wide entities identified during the Design phase are created in the Framework section of the application. Only when the system contains these framework entities is it possible to proceed with the creation of Contracts and their related Metrics.

Building the Framework entails adding the following new items:

- Servicess
- Service Groups
- Service Domains and Domain Categories
- Measurement Units
- Timeslot Templates
- Contract Parties
- Custom Attributes

For further information about each of the above items, refer to the Online Help.

Setting Up the Template Library (Contract Manager)

Template Libraries allows defining and managing:

- Template Libraries
- Template Folders
- Service Level Templates
- Contract Templates
- Security settings for user access rights

For further information about each of the above items, refer to the Online Help.

How to Create Contracts (Contract Manager)

At this stage, the Contracts and their related entities that were defined in the Design phase are created in the system.

Follow these steps:

1. Add new Contract and apply the Contract's general details.
2. For each Contract, define its Metrics and apply their general details.

Only the general details of a Contract's content is applied at this stage, without including the Business Logic and clustering of the Contract's Metrics.

The following description of these steps emphasizes some important points that should be considered at this stage. These steps are described in full detail in the Online Help.

Step 1: Add a new Contract and apply the Contract's general details.

The Contract definition should include:

- Setting the Contract's name.
- Selecting the related Contract Party(s).
- Attaching the related Services.
- Setting the effective dates for the Contract. The effective dates of the contracts are the date range for which the Correlation engine will calculate the service level for this contract; results for reporting are available only for these dates. When setting the dates, it is important to take into account the requirements in terms of the availability of the Contract related reports and the available raw data.
- Setting the Contract Time Zone and Currency. This definition is for reporting purposes and allows for the reporting on this Contract to follow the relevant Time Zone. The currency definition allows the Reporting engine to determine in which currency to display the penalty results for Penalty formulas.

Step 2: Add Metrics-general Details.

Once the Contract is in place it is time to create the Metrics within it. In the process of defining the Metrics, the following steps must be performed:

- Setting the Metric name.
- Applying Metric details (Service, Domain, Unit, Timeslot, Timezone, and so on).
- Setting Dashboard thresholds (refer to Configuring Dashboard Pages (see page 190)).
- Attaching related Metrics (if applicable) and the relationship between them.
- Defining the granularity at which the metric is calculated by the engine.
- Setting the objective statement and parameters.

At this stage, the Metric definition still lacks the following items:

- Business Logic formula/module and registration
- Clustering definition

These items are included in the Contract's Metrics only after the system infrastructure has been built and the Business Logic formulas have been developed and tested.

Note: An alternative method to the above approach s to first develop the service level templates within the system, rather than defining the contracts straight away. This enables creating the template which can then be used to create further contracts. In some cases, pre-existing service level templates can be imported into the system from another CA instance. This may allow a head start compared with creating the contracts from scratch. For more details on creating your own service level templates, refer to Creating Service Level Templates (see page 66).

In some cases, it is advisable to create a sample contract initially, to test that everything is working correctly and as expected. It is then possible to create the Level Template(s) from this contract and store it in the service catalog, providing a solid starting point for all of the contracts in the system.

Create Contracts from a Service

Creating contracts in the system using a Service Catalog, either from a template or without a template (based on multiple service level templates) provides much greater efficiency and high level of re-usability, applicable to many different contracts. In general, service level templates contain a collection of pre-defined Metrics applicable to certain service components. If required, you can also have more than one service (and associated Metrics) in a single service level template. Generally a service level template's contents are defined by how it will be used, and can vary according to requirements.

As an example, consider an Application Hosting service which is offered by an organization. The organization may offer its customers three different service packages such as Bronze, Silver and Gold, depending on what is included in the package. A good example of the use of service level templates could be to create one for each of the packages.

Once defined, these definitions can be used to create a new customer contract very efficiently. For example, Customer ABC decides to sign up for a Gold Application Hosting Package. You can create this new contract in the system directly using the service level template as follows:

In the Contracts page, click Add New, Using Service Catalog, and select either Based on Template or Without Template. Then follow the Contract Wizard to complete contract creation. If you choose Based on Template, then you need to specify the template settings.

The Contract Wizard displays a list of service level templates and you can specify which service level templates you would like to include in the Contract. Note that it is possible to select specific Metrics from multiple service level templates, or just select the whole definition for inclusion. In this example, all Metrics from the Gold hosting package. Note that selecting the top level automatically selects all the child nodes. Also note that it is possible to assign the Metrics to a different service if required. However, the default is to keep them assigned to the same service components as in the definition.

Once all of the required Metrics are selected, click the Next button to transfer those Metrics across to the new contract template and prompt for a contract name and general details to be entered. Click Save and Next to create the Contract.

Once this is completed you have the following options:

- Continue with the Contract Wizard, define parameters, and run the Metric Wizard, which opens the Wizard interface and allows customization of the contracts Metrics. The wizard allows each of the Metrics to be reviewed and modified by changing the available fields, such as the Metric Parameters in the objective statement, the Metric Name, Timeslot, and description. Once completed for each Metric, the wizard returns you to the Contract Metrics page, where you can close and save the new contract.
- Open the Contract page, to view or edit the contract.

Create Service Level Templates

Creating a service level template is a fairly simple process. From an existing Contract (on the Contract Details page) select all of the Metrics that you would like to include and click Save as service level template.

The next window requires you to name the service level template. You can then save the service level template. When saved, all Timeslots associated with the selected Metrics are included in the Timeslot tab. From here it may be necessary to further customize the service level template in order to make it fully flexible. This could include adding parameters to the Metrics and exposing these parameters via the Objective Statement for each Metric. Also perhaps the creation of service level template level Parameters (similar to Contract Parameters) which can then be referenced by some or all of the Metrics. Once completed, the service level template is available for deploying to other contracts.

Contract Life Cycle and Versioning

Once a Contract has been configured completely, it must be committed. This action allows the Calculation engine to begin calculating the service levels for all Metrics in the Contract. Committing the Contract changes its status from 'Preliminary' (where it is editable) to 'Effective' (where it is not editable). Any further changes required on this Contract require a new version of the Contract to be created. If the same effective dates are chosen for the new version, then after the changes are complete and the new version is committed, it completely overwrites the previous version. This also triggers the engine to recalculate the Metrics which are different to the previous version. Effective versions can also partially overlap, for example when a change is made to the targets of some Metrics in the Contract half-way through the effective dates. In this case, the old version is still used until the effective date of the second version becomes active. At this time, the second version assumes the effective status for the calculations.

The following table shows user changes against the recalculation impact scope and time frame. A change can impact a specific Metric within a contract-specific version or it can impact Metrics that are cross-contract and contract versions.

Change	Impact scope	Impact time frame
Changes Made in Metrics		
Changing Metric details - business logic formula	Impact all Metrics within contract-specific version	From beginning of contract effective version
Changing Metric details - target value	Impact all Metrics within contract-specific version	From beginning of contract effective version
Changing Metric details - tracking period	Impact all Metrics within contract-specific version	From beginning of contract effective version

Changing Metric details - Metric parameters	Impact all Metrics within contract-specific version	From beginning of contract effective version
Changing Metric details - time zone	Impact all Metrics within contract-specific version	From the beginning of the contract effective version
Changing Metric details - clustering	Impact all Metrics within contract-specific version	From beginning of contract effective version
Changing contract details -effective dates	Impact all Metrics within contract-specific version	From beginning of contract effective version
Changing contract details - timeslots	Impact all Metrics within contract-specific version	From beginning of contract effective version
Changing contract level parameters	Impact all Metrics within contract-specific version	From beginning of contract effective version
Changing SLALOM modules	Impact all Metrics attached to changed slalom module, in all contracts and contract versions	From beginning of contract effective version
General Operations		
Changing Metric resource model / Change set (CA Business Service Insight 4.0)	Impact all Metrics that register resource, in all contracts and contract versions	From closest state before date in which change in resource occurred
Getting delayed events for the Metric Events with past timestamp (raw data or reusability events)	Impact all Metrics that use resource, in all contracts and contract versions	From closest state before date in which change in event occurred
Add Metric corrections data	Impact all Metrics that use resource, in all contracts and contract versions	From closest state before date in which change in correction occurred
Change, update or delete Metric exceptions time (activation & deactivation)	Depending on exception settings and specific implementation, impacts specific Metric within contract specific version or can impact Metrics that are cross-contract and contract versions	As close as possible to exception time

Custom attribute update	Impact all Metrics that register resource, in all contracts and contract versions	From closest state before date in which change in resource occurred

Finally, some key points to remember about Contract versions:

- If the new version has the same effective date, only the Metrics that were changed are recalculated, and are recalculated from the beginning of the version.
- If the new version has different dates, all the Metrics in the new version are calculated from the beginning of that version and all the Metrics in the previous version are recalculated from some point in that version until the new version becomes valid. The exact amount of recalculation depends on the configuration of states.
- It is recommended to create contracts with effective dates for 1 year and renew them when they expire. This prevents recalculation periods longer than one year.
- Not effective versions (the current date is past the end of the effective contract dates) of contracts are calculated (and hence still count as active Metrics in the system, since they have calculated service level data associated with them for reporting).

The values associated with global variables within Metrics are not carried between contract versions (that is, the OnLoad routine in the Business Logic is called at the beginning of each contract version).

Note: For a number of walk-through scenarios and case studies refer to Contract Modelling Case Studies (see page 213).

Data Gathering (Data Source Expert)

In the data gathering phase of the implemenation process you work with adapters. The following topics outline that process.

Adapter Functionality

Adapters are modules in charge of collecting data from data sources and passing it to the CA Business Service Insight system. Adapters filter data coming from the data source and manipulate it in such a way, so that when it reaches the system it contains only the information needed for service level calculations in the correct structure.

The Adapter platform provides the flexibility to:

- Receive data online or offline with any required frequency
- Receive data at various levels; raw, calculated, or aggregated
- Receive data from a wide range of tool types

Basically, every Adapter consists of two components:

- **Generic Adapter component**:

  There are two types of the generic Adapter component, an ASCII file Adapter component and an ODBC based SQL Adapter component. These components allow connecting to a data source and parsing it as an ASCII file or executing an SQL query on it.

- **Adapter configuration file**:

  Every Adapter requires a configuration file to know where and how to connect, what to retrieve, and how to transform and translate the data into generic CA Transactions and Events. CA Business Service Insight provides every generic Adapter type with a default XML configuration template file which can be fine-tuned to represent customer specifics regarding the data source with which it has to connect. The XML configuration file defines which fields should be retrieved, how they should be identified, how they should be converted to the system normalized database, and so on.

  Note: An Adapter Wizard is built into the user interface allowing basic customization of this XML template online. This serves the same purpose as creating the XML configuration file for the Adapter. More details on this feature can be found later in this chapter.

The Adapter platform includes a restart/recovery mechanism and can handle problems in data received from third party tools, such as tool crash, network problems, missing data, duplicate data, garbage data, gaps in the data, data validation, and so on. Every Adapter provides for full data integrity and complete tracking and logging of all Adapter messages and is covered later in this guide in more detail.

CA Business Service Insight Adapters can run as a service or as an application (visible or not visible). CA Business Service Insight Adapter technology supports advanced security mechanisms, such as, encryption, hand shake and authentication processes.

It is important to note at this stage, that the Adapter Wizard is a mechanism of automating the following processes and tasks. While certain elements mentioned may not always be visible when using the Wizard-driven approach, they are all still present “behind the scenes” of the wizard interface.

Adapter Environment

The following entities relate to the Adapter and its configuration and execution parameters.

Data source**:**

Data source to which the Adapter connects and from which it retrieves data in its original format.

Work files**:**

Output files produced by the Adapter and which are written within the processes (for more details refer to Work Files (see page 73)).

CA Business Service Insight Adapter listener**:**

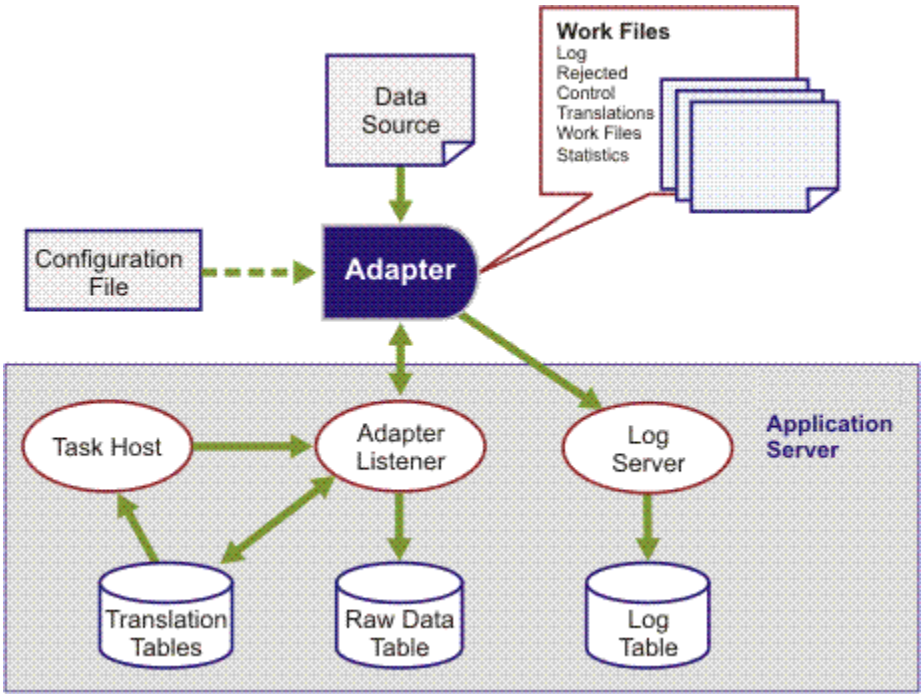
Three message types are transferred between the Adapter and the Adapter Listener:

- **Control**: Start\stop\pause messages sent by the listener to the Adapter and back again when the Adapter changes its status.
- **Translations**: Adapter sends requests for the translation table content and requests for specific translation values. The listener returns the tables and the translated value. The Adapter listener receives the indication from the Task Host that a translation entry was translated. It then sends the message to the Adapter.
- **Raw data**: Unified raw data events sent by the Adapter. These events are sent in packets and include acknowledgments.

CA Business Service Insight Log server

Adapter can be configured to send log messages to the system log as well as writing them to a local file. If the port and IP address of the log server are specified and set in the registry settings of the Adapter, then the Adapter sends messages to the log server as well.

The following diagram describes the Adapter process in relation to each of the entities with which it interacts.



The following is a description of the Adapter process interaction with these entities:

Configuration file**:**

Contains settings for some or all of the Adapter's configuration parameters. The Adapter uses the configuration file to determine the connection method that is used by the Adapter and the Metrics for parsing in order to create the Event output. This is an XML file, and the format contains six basic elements:

General:

Miscellaneous Adapter attributes (working directory, output files, debugging flag).

OblicoreInterface:

Attributes for connection with the CA Business Service Insight Server.

DataSourceInterface:

Attributes for connection with the data source (files path and pattern, connection strings, SQL queries, and so on)

InputFormatCollection:

Parsing Metrics for analyzing and manipulating source data.

TranslatorCollection:

Metrics for building the unified event compounded of the parsed and manipulated data fields.

TranslationTableCollection:

Metrics for mapping data between original data and CA Business Service Insight entities.

Each of these six sections contains all of the relevant information that allows the Adapter to connect to the data source, retrieve the required information, parse it into CA Business Service Insight unified events structures, and store it within the CA Business Service Insight Raw Data table.

Main Files

The Adapter consists of two main files: the executable and configuration files. The executable file is a generic file. There are two such executable files: SQL Adapter and file Adapter.

An XML configuration file is tailored for each Adapter in order to store the specific data source requirements. The configuration file specifies the information related to the data source (name, location, connection method and structure) and the structure of the output Events that are to be generated by the Adapter.

The Configuration file includes those parameters and values that are set for attributes within a pre-defined structured XML file.

When creating a new Adapter, it is necessary to use the existing relevant executable (according to the target data source type, file for flat file data sources, SQL for database data sources) and then modify the configuration file as needed. The two structures contain slightly different configuration elements for a text or SQL Adapter. This is generally set up automatically when creating the Adapter using the Adapters Manager utility.

Other files related to Adapters are work files created by the Adapter in the process of reading the data source and writing events to the CA Business Service Insight system.

Note: For details on modifying the Configuration file, refer to Adapter Configuration Specifications (see page 311).

Work Files

The Adapter work files are created when the Adapter runs for the first time and are updated constantly during every Adapter run.

Each Adapter has its own work files. The names of the work files can be set in the Adapter configuration file (optional) or may retain their default names. The location of the work file is set by the "Working folder" and can also be set in the configuration file. Note that the path specified may be relative to the current directory within which the Adapter is located. The path specified must already exist (or you must create it) for the Adapter to execute correctly.

Note: The folder is not created automatically if it does not exist.

All relevant parameters in the configuration file are contained in the General section. Only the location of the log file is set in the registry or passed via the command line.

AdapterStatistics.txt

The Adapter writes statistic information to this file at one minute intervals. The last line is written when the Adapter stops. Each line in the file contains numbers of:

- Received events
- Ignored events
- Events with errors
- Sent messages
- Sent packages

Each time the Adapter runs it initiates the statistics.

RejectedEvents.txt

This file contains all Events that the Adapter failed to send to CA Business Service Insight because the Event value, defined as one requiring translation, has no matching ID in the translation table. This means that the relevant translation was not performed. Each Event that has at least one value waiting for translation is written to the rejectedEvents file.

At the beginning of every run, the Adapter first attempts to send CA Business Service Insight the Events from the rejectedEvents file as it tries to find the matching ID for the relevant value in the translation table. If the value is found, the Adapter sends the Event and deletes it from the file. If a matching value is not found, the Event remains in the rejectedEvents file.

It is possible to configure the upper limit for the number of rejected Events that can be reached by setting the RejectedEventsUpperLimit parameter in the Adapter configuration file. When the limit is reached, the Adapter stops reading new records and enters a 'Blocked' status. This can be seen when the debug output is displayed on screen during the Adapter execution. If you see a continuous string of uppercase B's, then the Adapter is blocked and requires some of the pending entries to be translated before it loads any more data.

The Pending Events are written to the file in an XML format. Following is an example of an Event from the file:

```
<rejectedEvent
        createDate="1062330841"
        translator="Translator">
        <event
        inputFormat="InputFormat">
                <field name="resource" type="3" value="Server333p"/>
                <field name="timestamp" type="4" value="1036108800"/>
                <field name="memory_utilization" type="2" value="26.71"/>
                <field name="cpu_utilization" type="2" value="78.85"/>
        </event>
</rejectedEvent>
```

Adapter log

The Adapter log is the file to which the Adapter writes log messages.

It is recommended browsing the Adapter log file using the Log Browser utility.

It is possible to set the level of reporting to this log file by modifying a parameter in the Adapter configuration file-LogDebugMode. When set to "yes", the Adapter writes normal indication messages to the log, as well as original record, parsing results, and destined Event.

This parameter is usually set to "yes" during Adapter testing and monitoring.

By default ,the file size is limited to 1MB. When this limit is reached, the Adapter changes its name and append "_old" and create a new log file. The Adapter can potentially store up to 2MB of log messages; 1MB for the old file and 1MB for the current file.

The size limit of the log file can be configured as an entry in the Registry for each Adapter with a maximum of 10MB. The entry in the registry is named LogFileMaxSize and is defined under the specific Adapter with its value being a multiple of KB.

DataSourceControl.xml

The DataSourceControl.xml file is used by the Adapter to control its access to the data source and ensure that whenever it executes, it always continues onwards from the last point to which it read.

The file Adapter retains the name of the last file read, the last read line, and the position in the file it reached. The next time the Adapter runs, it accesses the file from the location using the information found in the DataSourceControl.xml file. Using this mechanism, the Adapter can read only new records on each run.

The Adapter does not work directly on the source files, but first copies the current file to a work file. Hence, the same information is kept for the work file and for the source file. If the source file is appended, only new records are copied to the work file.

If the Adapter is configured to delete the source file once processed, by setting the parameter in the configuration file DeleteAfterProcessing to "yes", it does not save the information in the source file. When finished, it reads any new file that exists in the working folder that matches the file pattern defined in the configuration file.

Only when DeleteAfterProcessing is set to "no" does it check for new records in the last file. If there are none, it moves to the next file in a lexicographical order. Therefore, when naming source data files, to ensure they are read in the correct sequence, try and enforce an increasing sequential naming pattern. For example, append the files with a reverse date value (yyyymmdd-hhmmss) to support this. For example:

- DataSourceABC20070517-14:00:00.csv
- DataSourceABC20070517-15:30:00.csv
- DataSourceABC20070517-17:00:00.csv and so on….

The following is an example of the DataSourceControl.xml for a file Adapter.

```
<AdapterControl Save="end" LastSaveTime="2005/05/20 13:06:39">
<Data><WorkData><LineNumber>0</LineNumber>
<FilePattern>c:\adapters\callsadapter\*adapterpca.csv</FilePattern>
[set the File Name variable]</FileName>
<BasicPosition>0</BasicPosition>
</WorkData>
<NonDeletedFiles>
<File NamePattern="c:\adapters\callsadapter\*adapterpca.csv">
[set the File Name variable]2005adapterpca.csv</FileName>
<LastLine>25/04/2005,5925,NN4B,12,12,0,10,0,11</LastLine>
<LastPosition>15427</LastPosition></File>
</NonDeletedFiles>
</Data>
</AdapterControl>
```

A SQL Adapter retains the last value of the query key fields for each query executed. The key fields are unique identifiers for records in the destination database table. The Adapter uses these values when building the query for the next run. This allows the Adapter to read only new records.

For example, consider the following SQL statement used for fetching some Trouble ticket data.

```
Select ticket_id, status, organization, open_date, respond_date, resolved_date,
record_modified_date from t_ticket_data;
```

In this example, to capture all the latest records from the data source, the query key field chosen to ensure obtaining the latest information is record_modified_date, since it generates new tickets raised since the last Adapter execution, as well as updates to existing tickets. By selecting this field as the query key field, the Adapter automatically appends the following section to the end of the query during execution:

```
where record_modified_date > :previous_value order by record_modified_date asc
```

Therefore, it only retrieves the newer records. Note that there are a number of considerations when choosing the query key field, and they always depend on the behavior of the data source and what you are trying to achieve with the retrieved data. Also note that the fields chosen in the previous example are not always the best choice for *every* situation.

An example of the DataSourceControl.xml file for a SQL Adapter is shown below.

```
<AdapterControl Save="end" LastSaveTime="2005/05/20 15:59:02">
<Data><QueryCollection>

<Query QueryName="ChangeManagementOpenQuery">
<KeyField Name="Incident Ref"><LastValue>32357</LastValue></KeyField>
<KeyField Name="Date Logged"><LastValue>18/04/2005 16:56:26</LastValue></KeyField>
<LastFileName/>
</Query>

<Query QueryName="ChangeManagementPendingQuery">

<KeyField Name="Incident Ref"><LastValue>0</LastValue></KeyField>
<KeyField Name="Date
Resolved"><LastValue>1900-01-01</LastValue></KeyField><LastFileName/>

</Query>
```

send.txt

All Events that are built and are ready to be sent to CA Business Service Insight are first written to the send file.

SendControl.xml

The sendcontrol.xml file contains all lines that were sent and acknowledged by CA.

The file allows the Adapter to follow the sliding window acknowledgment protocol for transferring the data to CA. Additional information on this mechanism can be found in Adapter Communication (see page 80).

```
<SendControl
PacketMaxSize="50"
LastAckSequence="47"
LastAckIndex="-1"/>
```

IllegalEvents output file (.txt)

The Adapter writes to the IllegalEvents output file all records that were read, but had an error parsing them. This is commonly caused by validation logic entered into the adapter configuration file. The Adapter saves these records if the parameter SaveIllegalEvents in the configuration file is set to "yes". Note that the path for this option must also be set by using the 'IllegalEventsDirectoryName'. This folder must already exist, since it is not created automatically. If the folder does not exist, the Adapter gives an error when executing.

Within a file Adapter, the file containing the error records have the same name as the source file, whereas in the SQL Adapter, it bears the name of the query.

Translations.xml

The translation.xml file contains the Adapter translation tables.

When the Adapter runs in online mode, the file contains a copy of the translation table from the database. If the translation table is configured as remote, the Adapter loads the translation table from the database into this file, replacing it. If standalone, it reads the local file.

When the Adapter runs in Offline mode, it uses the file only as its translation table (for additional information on Online/Offline modes, refer to Adapter Running Modes (see page 85))

```
<TranslationTableCollection>

<AssystResourceTable>
<Entry TranslationStatus="Ignored" resource="Authority"/>
<Entry TranslationStatus="Translated" resource="LONDON" TranslationTo="1006"/>

</AssystResourceTable>

<AssystSupplierEventTypeTable>
<Entry TranslationStatus="Translated" severity="1" TranslationTo="1545"/>
<Entry TranslationStatus="Translated" severity="2" TranslationTo="1550"/>
<Entry TranslationStatus="Translated" severity="3" TranslationTo="1551"/>

</AssystSupplierEventTypeTable></TranslationTableCollection>
```

Adapter Communication

Adapters interact with the data source on the one hand and the CA Business Service Insight Adapter Listener and log server on the other, as depicted in the following diagram.

The Adapter communicates with the data source to retrieve the data using an ODBC connection and may be located either locally or remotely from the data source, as long as the Adapter can establish the ODBC connection.

The Adapter communicates with the CA Business Service Insight application server using TCP/IP protocol and can therefore be located locally or remotely from it, as long as it can establish the TCP/IP connection.

The Adapter must have two ports open, one for the Adapter Listener and one for the log server. The Adapter listener ports must be unique per Adapter and should not conflict with other network operations or applications that may also be using these ports. For example, you should not use Port 1521, since generally this port is used by the Oracle TNS protocol for communication to the database, and so on. You may also need to consider any local firewalls which may block this traffic.

Note: Check with your local administrator if you are not sure about which ports are available for use, or if you need to request ports to be opened to allow this communication to take place.

The port and address of the Adapter Listener is set in the Adapter configuration file. The port and IP address of the log server is set via the Adapter's entries in the registry.

The client/server operation with regard to the Adapter Listener is configurable, which makes it possible to configure the Adapter to operate as a client or as a server. The configuration of the client/server operation is done on the Adapter side in the configuration file's parameters. To do so the Port, Address and ConnectionInitiator variables must be set accordingly.

If the ConnectionInitiator is set to be the Adapter, then only a destination port is required. If it is set to be CA Business Service Insight, then a port and an IP address of the Adapter Listener on CA Business Service Insight is required. By default, the server is set to be the Adapter. This is sometimes an important feature to enable a firewall rule to be triggered (a feature known as port triggering). Sometimes a firewall only allows an inward request on a port, if a message was sent from the 'inside' of the firewall on that same port. It then triggers the firewall to allow communication to take place.

Note: Consult your network administrator for more information about local conditions that may affect Adapter communications.

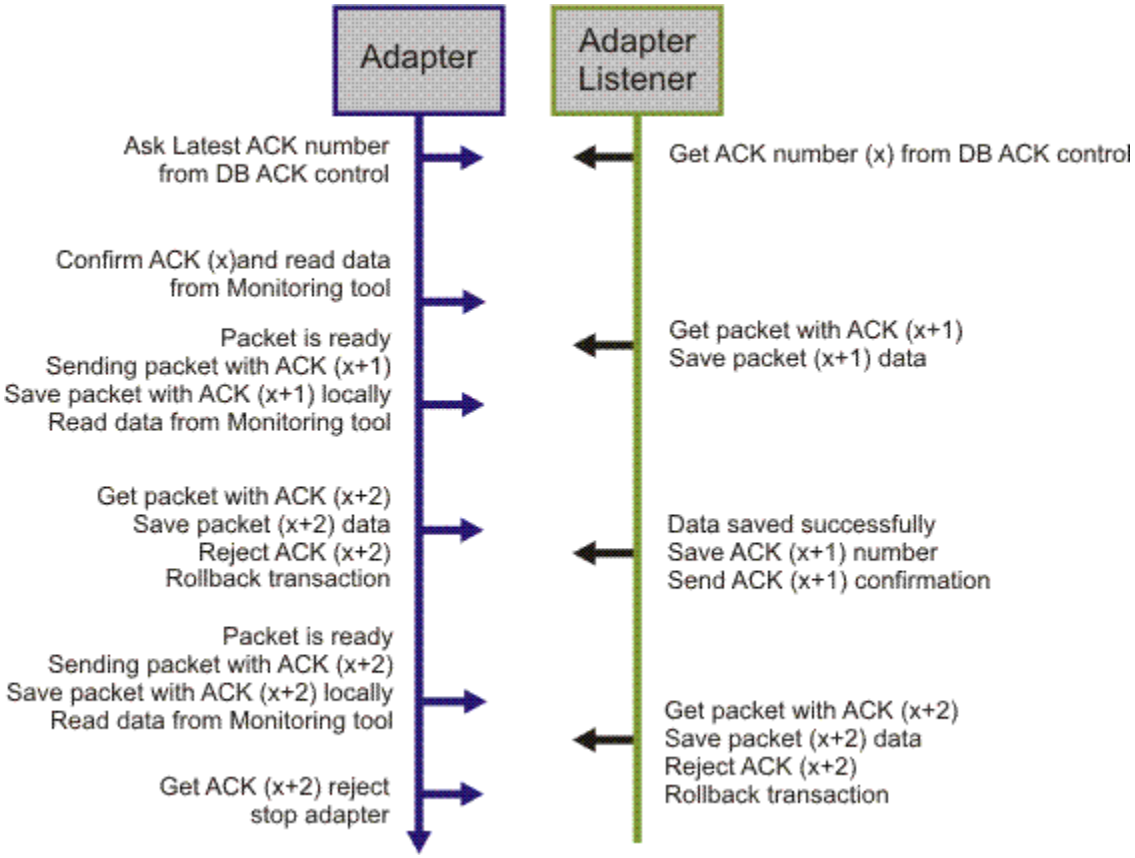
From a security point of view, it is recommended that the Adapter is set to be the client since this ensures the destination of Events when working in a multiple deployment environment for testing and production.

To verify transmission success of data records from the Adapter to the CA Business Service Insight Adapter listener, the Adapter incorporates an ACKs/sliding window algorithm over the TCP/IP layer. This algorithm basically sends the data in packets and then waits for an acknowledgment from the Adapter listener before moving to the next packet. Each packet contains several raw data messages. The number of messages in a packet can be configured by setting the Packet Size parameter. Each packet has a sequence which is contained in the acknowledge message. All relevant parameters that control the process are contained in the CA Business Service Insight interface section of the configuration file. In general however, you do not need to alter these parameters.

The Adapter's Listener writes the raw data in the packet in a single transaction.

Note: The ACK operation can be done only on the raw data messages sent to CA Business Service Insight.

The following figure shows the Adapter communication process.



Adapter Registry Settings

In cases where the information is missing from the command line, the Adapter uses some of the definitions stored in the system registry on the server where the Adapter is installed.

The registry entries are written by the Adapter Manager utility if the utility was used for the Adapter installation. If it was not used for the Adapter installation, these entries can be added manually to the registry.

Note: If installing an Adapter in a UNIX environment, these entries must be added manually since there is no Adapter Manager for this environment.

Listed below are registry entries used by the Adapter and the Adapter Manager utility.

General Server Entries

The following entries are written in the \HKEY_LOCAL_MACHINE\SOFTWARE\Oblicore\Adapters registry:

Possible properties:

Name	Type	Description
AdaptersDir	String	Root directory for all Adapters.*
FileAdapterConfTemplate	String	File Adapter configuration template path.* Adapter Manager uses this information to copy configuration template to new Adapter's folder where specified to do so, as part of Adapter's creation process.
GenericFileAdapter	String	File Adapter executable.* Adapter manager either creates a shortcut to executable or copies it to new Adapter's folder as specified to do so, as part of Adapter's creation process.
GenericSqlAdapter	String	SQL Adapter executable.* Adapter manager either creates a shortcut to executable or copies it to new Adapter's folder as specified to do so, as part of Adapter's creation process.
LogServerAddress	String	Log Server Network address. (Optional)** Log Server port-usually 4040. (Optional)** In cases where these parameters are set, Adapter reports log messages to the CA Business Service Insight log server.

Name	Type	Description
LogServerPort	String	
SqlAdapterConfTemplate	String	SQL Adapter configuration template path.* Adapter Manager uses this information to copy configuration template to new Adapter's folder where specified to do so, as part of Adapter's creation process.

* Used only by Adapter Manager utility

** Used by Adapter

Individual Adapter Entries

The following entries are written in the \HKEY_LOCAL_MACHINE\SOFTWARE\Oblicore\Adapters\<Adapter Name> registry:

Possible properties:

Name	Type	Description
ConfigurationFileName	String	Adapter configuration file name.**
Directory	String	Adapter directory*
LogFileName	String	Adapter log file name.**
Path	String	Adapter executable path*
RunAs	Number	Running mode.* Service/Console Application/Windows Application
Type	Number	Adapter type.* File/SQL
LogFileMaxSize	Number	Value is a number in KB.** Allowed range is 1,000-100,000 and default value is 1000.

* Used only by Adapter Manager utility

** Used by Adapter

Adapter Running Modes

The Adapter can be executed by either:

Service**:**

Adapter can be installed as a regular Windows service. This enables the system to control its status (Run, Pause, Stop, Automatic) as it would a regular Windows Service.

Installing the Adapter as a Windows Service which is done by running the Adapter executable from the command line using the **-i** to install it as service and **-u** to uninstall it.

Application**:**

Running the Adapter executable from the command line. The Adapter command line can be run in the following manner:

Command line options:

```
TextFileAdapter.exe -i | -u | -v | -d [-t] [-f configurationFileName] [-l
logFileName] [-n serviceName]
[-a OblicoreAddress]  [-p OblicorePort] [-la LogGerheaDed] [-lp LogServerPort]
```

Parameter	Function
-i	Install service
-u	Remove service
-v	Show version
-d	Run Adapter as console application
-t	Check only-check configuration file and stop
-f	Set configuration file name
-l	Set log file name
-n	Set service name
-a	Set application server address
-p	Set application server port number (1024-49151)
-la	Set log server address
-lp	Set log server port number (1024-49151)

This type of execution is commonly used in projects. This allows the execution of the Adapter via a .bat file and also allows using the Windows Scheduler to control the timing of the Adapter execution. To schedule the Adapter using the Windows Scheduler, it is necessary to configure its running mode to be Run Once.

RunOnce: (optional [yes/no]). When set in the configuration file to "no", once executed the Adapter runs continuously. When set to "yes" the file Adapter runs, reads records and stops automatically when no new records appear. A file Adapter reads the whole file, then waits a few seconds and tries to read new records (depending on the SleepTime settings). If there are no new records, it stops. An SQL Adapter runs each one of the queries only once. If RepeatUntilDry is set to "no", it stops immediately. If RepeatUntilDry is set to "yes", it waits (depending on the SleepTime). It tries to run the query again (depending on the query's sleep time), and if there are no new records, it then stops.

For details on the SleepTime and RepeatUntilDry attributes, refer to Adapter Configuration Specifications (see page 311).

The CA Business Service Insight Interface section of the configuration file consists of attributes specifying the two connection modes to CA Business Service Insight: online and offline.

In online mode, the Adapter connects to CA Business Service Insight, retrieves the translation tables and control commands from CA Business Service Insight, then sends Events, statuses, and translation requests back to CA Business Service Insight. In offline mode, the Adapter works with a local translation table file and writes the Events to an output file.

Offline mode is commonly used when first developing an Adapter and testing it.

Using ConsoleDebugMode set to "yes" allows debug messages to appear on the console.

For details on the various indicators, refer to Adapter Configuration Specifications (see page 311), specifically the ConsoleDebugMode attribute.

Configuration and Maintenance Tools

The tools required as part of the process of configuring Adapters and maintaining them are mainly standalone utilities, such as the CA Business Service Insight utilities which are simple Windows executables.

Note: When configuring Adapters in a UNIX environment, these utilities are not available, and configuration must be performed manually.

When working on an CA Business Service Insight application server, the utilities are installed as part of the application installation and are located in Program Files\CA\Cloud Insight\Utilities folder.

As part of this installation, a shortcut is created from the Start menu. It is recommended to use this shortcut to run the utilities.

In a situation where one is not working on the CA Business Service Insight application server, these utilities can be copied to any Windows computer as standard files, or can be installed using the supplied CA Business Service Insight package and choosing a custom installation. The installation routine is not mandatory however, and coping the.exe files to a convenient location on the workstation is usually sufficient. Using this option however, you may find some .dll files are missing and should also be copied, if available, from the server to the local folder on the workstation.

How to Configure Adapters and Translations

This stage includes performing the following steps:

1. Configuring the Adapter using the Adapter Wizard or manually editing the XML configuration file (described in the following chapter).
2. Deploying the Adapter.
3. Testing the Adapter.
4. Performing Translations.
5. Writing Translation scripts in order to support an automatic translation process (optional).

Note: Deploying the adapter can be done automatically when using the Adapter Wizard as the new Adapter Deployment service runs as a background service on the application server to handle this task.

Deploy a New Adapter (Adapter Wizard)

When you create a new adapter using the adapter wizard, ensure the “Adapter Listener” and “Adapter Deployment” services are running.

Deploy a New Adapter (Manually)

Prerequisites for Creating a New Adapter

All of the following need to be in place before starting to create a new Adapter:

- **Adapter's root folder**: If CA Business Service Insight is installed on the server, this folder exists under the Program Files\CA\Cloud Insight root folder, if not, it should then be created.
- **Individual Adapter folder**: Create a folder under the Adapter's root folder for the specific Adapter.

  Note: If you are using the AdapterManager utility, the utility automatically creates the folder when adding the new Adapter.
- **Adapter's executables**: TextFileAdapter.exe, the text file adapter's executable for text file Adapter; SQLAdapter.exe, SQL file adapter's executable for the SQL Adapter. These can be found on the application server under the Program Files\CA\Cloud Insight\Adapters folder.

  Note: During the Adapter creation process via the Adapter Manager utility, whenever possible, you should choose the option to create a shortcut to the executables, rather than create a copy. This ensures that when an upgrade or service release (SR) is applied to CA Business Service Insight all of the binary components are updated correctly.
- **Configuration Templates**: Templates of the Adapter configuration file. Place the files in the Adapter's root folder. These can be found on the application server under the Program Files\CA\Cloud Insight\Adapters folder. The configuration templates are used to create the configuration file to avoid having to create them from scratch. It is also common to use an existing Adapter configuration file for this purpose.
- **Adapter Manager utility**: A standalone executable. It is sufficient to make a copy of the AdaptersManager.exe from the Utilities folder under the Program Files\CA\Cloud Insight\Utilities folder on the application server. It is not necessary to create the Adapter using this utility, and this utility can be used only on a Windows server.
- **Two open TCP\IP ports**: One port for the AdapterListener on the Application server and another for the LogServer. The LogServer port is usually 4040.
- Check which CA Business Service Insight application service components are running. For the purposes of running the Adapter, the following service components need to be running:
  - **AdapterListener**
  - **TaskHost**
  - **LogServer**

Follow these steps:

1. Run the AdapterManager Utility (see AdapterManager Utility section above).
2. Prepare all the above prerequisites and create a batch file with the executable command line (see Adapter Running Modes (see page 85)).

Modify the Adapter Configuration File

When creating an Adapter, the bulk of the work is editing the configuration file.

This work involves setting the attributes in the XML file to control the behavior of the Adapter so that it performs as required. The configuration file is an XML file with each section corresponding to a step in its internal work-flow.

The sections are:

- **General section**: Miscellaneous Adapter attributes, working directory, output files specifications; debugging flag, defaults, and so on.

Oblicore:

- **OblicoreInterface section**: Attributes of the connection with the CA Business Service Insight Server (TCP/IP port, security mode, and so on).
- **DatasourceInterface section**: Attributes of the connection with the data source (files path and pattern, connection strings, SQL queries, and so on).
- **InputFormatCollection section**: Parsing rules for analyzing and manipulating the original data format (delimiters, field types, order of data, regular expressions, and so on).
- **TranslatorCollection section**: Rules for the CA Business Service Insight unified Event compounded from parsed and manipulated data fields.
- **TranslationTableCollection section**: Rules of data mapping between original data terminology and CA Business Service Insight data entities.

These sections are described in detail in Adapter Configuration Specifications (see page 311).

Note: The order of the XML nodes within each section is not important.

File Adapters

File Adapters use the FileAdapter generic component (executable) and configuration file with which to parse ASCII files.

File Adapter work-flow:

- Copy/rename the source file to a work file.
- Read the logical record.
- Parse the record according to delimiters or regular expressions.
- Find the right inputFormat.
- Build the Event record.
- Translate the Event record.
- Update control file.

Configuration File Example

The following example uses a simple ASCII file data source with its Event output requirements and reviews the settings required for its Adapter configuration file.

The configuration file can be viewed and edited using the XMLPad utility.

For an overview of the structure and content of the configuration file, refer to the relevant sections.

Note: The settings reviewed are only the main and most important attribute settings. For full attribute specifications see Adapter Configuration Specifications (see page 311).

File Adapter Case Study

Consider the following server monitoring system which produces log files with information according to the following structure:

Server	Timestamp	Memory Utilization	CPU Usage
Server001p	1.11.2004 0:00	26.71	78.85
Server002p	1.11.2004 1:00	49.09	89.49
Server001p	1.11.2004 2:00	51.37	57.51
Server001p	1.11.2004 3:00	23.53	98.9
Server002p	1.11.2004 4:00	86.21	85.11
Server002p	1.11.2004 5:00	39.98	35.97
Server001p	1.11.2004 6:00	70.58	55.96

The files are delivered to the CA Business Service Insight Adapter folder into a sub-folder called 'data'. The file names are all prefixed with 'ServerData' and suffixed with the date and time. The files are also CSV files with the extension of .csv.

The following output Event is required:

- **Resource translation field**: Server
- **Timestamp field**: Timestamp
- **Data fields**: Memory Utilization, CPU Usage

In addition, it is assumed that the data source is located in Belgium (CET time zone which has a +1 time zone offset, and includes daylight saving periods of an addition hour during summer time).

Configuration File General Section

- **MajorVersion And MinorVersion**: Should be set by default to the application version.
- **WorkingDirectoryName (optional)**: Specifies the default path for Adapter output files (data source control, translation, sends). In this case, it is set to "output", and as such this folder is then created under the Adapters main folder.

The following Indicators control the way the Adapter reads and translates the records:

- **RunOnce**: If set to "Yes", the Adapter executes once, reading all available data and then stops.
- **DataSourceDifferenceFromUTC**: Indicates the time difference between UTC (Universal Time Constant, always with zero time offset. Equivalent to GMT) and the time zone of the data source. The time zone of the data source is the time zone with which the time fields in it are denoted. The reason for having this attribute is because the Adapter normalizes all dates into UTC. Having all dates in UTC allows the application the flexibility to then display times according to the user's requirements. The following attributes provide the Adapter the details of how to transform the time fields from the input to UTC:
  - **DefaultOffset**: Offset from UTC when not in daylight saving time. In this case, it is set to '1' for central European time (CET).
  - **TimeFormat**: The format by which the Daylight Saving details (described next) should be parsed. The European time format is 'dd/mm/yyyy hh:mi:ss', whereas CA Business Service Insight format specifications are set as '%d/%m/%Y %H:%M:%S'.
  - **DayLightSaving**: One daylight saving period of the data source time zone. This element is optional (meaning, if not selected, there are no daylight saving times) and it can exist more than once; Once for each daylight savings period entered. When some elements exist, they must be ordered by time, and the periods must not overlap. Usually when configuring Adapters, this element is specified to be a number of years ahead. In this case, only one year is configured as an example.
- **From**: Begin date of the period. Specified using the time format set above '25/03/2007 02:00:00'.
- **To**: End date of the period. Specified using the time format set above '28/10/2007 03:00:00'.
- **Shift**: Time shift added to the DefaultOffset within daylight saving period. Enter '1' as the time shifts forward an hour during this daylight savings period specified between the 2 dates specified.

Configuration file OblicoreInterface Section

This section defines the connection to the CA Business Service Insight server.

- Mode-in online mode, the Adapter connects to CA Business Service Insight, retrieves the Translation tables and control command from CA Business Service Insight, then sends Events, statues and translation requests to CA Business Service Insight. Configure the adapter in this mode, and set the value to 'online'.
- Port-the TCP/IP port number the Adapter uses to communicate with the CA Business Service Insight server (where the AdapterListener is located).

  Assuming there are no issues with doing so, configure this adapter to use port '5555' (chosen arbitrarily). This must also be specified on the server in the GUI for the adapter too, to enable communication.

Configuration File DataSourceInterface Section

The DatasourceInterface section consists of attributes specifying the connection and the connection type between the Adapter and data source. There are two types of interface, file and SQL. The main difference between the two is that for files, the Files collection is required, and for SQL, the queries collection is required.

The DataSourceInterface section also defines the way in which the Adapter manages the source file (whether it deletes the original file if it was created only for the Adapter, or whether it leaves the data if it is needed for other uses, and so on).

For file Adapters, to read and parse ASCII files, the files interface is used as shown in the following figure. Choose the following values for the settings as follows:

The Files section under the DataSource Interface node relates to the connection to the data source. Configure the following attributes.

Note: This section looks completely different for an SQL adapter.

- **DeleteFileAfterProcessing**: Sets the way in which the Adapter handles the source file and determines how the Adapter controls the reading in order to read only new records. In this case, the source files are left in place on the server and this value is set to 'no'.

  In cases where a file is created only for the Adapter and it can be deleted after processing, it should be set it to "yes". The file is then renamed, processed, and deleted.

  When set to "no", the file is copied and the processing takes place using the copied file. If new records are appended to the end of this file, the Adapter copies these new records to the work file during the next cycle. If new records are not appended to the file, the Adapter searches for the next file with the same pattern and name (in lexicographic order) as the current file. If the Adapter finds such a file, work proceeds on this file. The Adapter does not revert to the previous file even if new records are appended.

  Set to "no" when you need to keep the integrity of the source file and when the file should be appended.

- **InputFormat**: Refers to the name given to the InputFormat element in the next InputFormatCollection which handles the data from this data source connection. The Input format is the fields structure of the input data coming from the data source after being parsed by the Adapter. The parsing method is specified in the delimiters attribute as explained below. When handling more than one connection using different interface formats this field becomes more important and determines which input format structure handles each data sources data.

- **Path**: Physical location of the data source files. For example : C:\Program Files\CA\Cloud Insight\Adapters\ServersAdapter\data\.

- **NamePattern**: Specifies the data source file name. Wildcards may be used if more than one file uses the same input format. If a file name is specified with no wildcards, the Adapter searches only for a file with this name. Where using wildcards, the Adapter searches for all files that correspond to the pattern, sorts them lexicographically, then reads them one by one. During the next run, it searches for new records in the last file before proceeding to the next.

  In this example, if the wildcard character '*' is used, then the attribute values is "ServerData*.csv". (The Adapter reads all files with names that begin with ServerData and have the extension, .csv.)

  Important! It is recommended that a date and time are added to the end of the filenames using the following format YYYYMMDD-HHMISS to ensure that the files are sorted correctly, read in the correct order, and that none of the files are missed. The Time portion can also be added if there are multiple files being produced each day.

- **Delimiters**: Method of determining how the file is parsed. One or more characters can be specified to serve as delimiters, according to the data rows that are to be parsed into fields. If not specified, the default value is "/t".

The data source file in this example is a CSV (comma separated) file. The simplest way of parsing such files is to specify the comma as a delimiter.

Other methods available for parsing are:

- **RegExForParser**: Uses a regular expression to set the parsing rule.
- **LogicLineDefinition**: Used when a line in the file is built out of several lines.
- **TitleExists** (optional): Specifies whether the first non-blank line in the data source file is a title line. The title can be used by the Adapter when parsing the data. In this example, each data file contains the title row, so 'yes' should be specified for this attribute.

Configuration File InputFormatCollection Section

This section specifies the structure of data retrieved from the data source-how a data row is to be cut into fields and what are the field types and formats. Initial data filtering and data manipulations may be performed in this section by using InputFormatSwitch and Compound fields respectively.

In this section it is possible to define the validation Metrics for the input records using the InputFormatValidation and the ValidationCase, which determine whether or not a record is legal.

The InputFormatCollection node may contain one or more InputFormat nodes.

The general work flow of this section that the Adapter follows is:

- Data row is matched against the InputFormat specified in the DataSourceInterface.
- Data is dissected into fields following the matching InputFormat specification. The order of the InputFormatFields should correspond with the order of fields parsed from the data source.
- Compound fields are assigned values by combining and breaking the previously specified data fields. Compound field definitions should come after the normal field definition.
- Processed data is checked against TranslatorSwitch conditions to determine which Translator is used to build the output event out of the input record.
- Processed data is either sent to the matching Translator or is ignored.

Work with the following parameters:

- **InputFormatName**: Name for this format, to be referred to by DatasourceInterface section.
- **InputFormatFields**: InputFormatFields may contain one or more Field nodes according to the data source number of input fields.
- **InputFormatField**: Specifies one data field of the original data row, or a compound field.

  <InputFormatField Name="timestamp" Type="time"

  TimeFormat="%d/%m/%Y %H:%M:%S"/>

  - **Name**: Name for this field, to be referred to by other elements, such as the compound element or the TranslatorFields as source field.
  - **Type**: Data type of the field-string/integer/real/time.
  - **Source**: Source value for this field, Default value taken is "event", possible values:
    - **event**: Field value is taken from the event coming from the data source. Values of fields are taken in the same order as coming from the data source.

- **compound**: Field value is built from other fields, based on a manipulation of other fields' values or constants. The fields manipulated should already be defined.
- **title**: Field value is taken from title field names. The referred field should already be defined.
- **filename**: Field value is taken from the data source filename; only for text file Adapters.
- **constant**: Field value is constant and taken from the ConstantValue whose property should appear next.

- **TranslatorSwitch**: Determines which Translator is used to translate the data row into a Unified Event.
  - **DefaultTranslator**: Translator to be used in cases where no criteria can be matched. If the value is set to "Ignore", no Translator is used and the line is ignored and not sent to CA Business Service Insight.

Configuration File TranslatorCollection Section

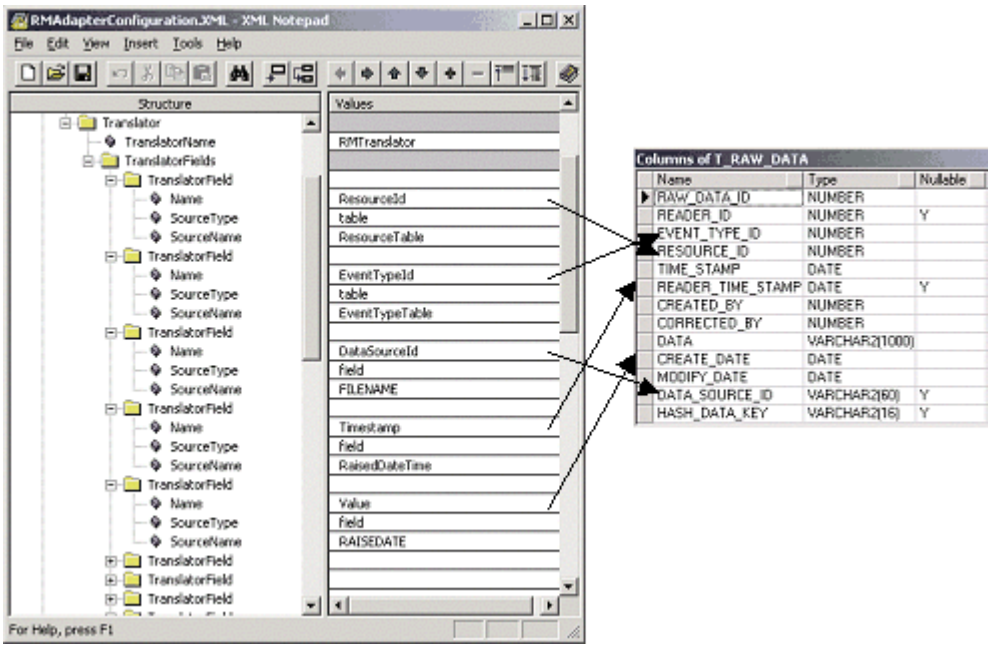
The Translator Collection section defines how the parsed and manipulated data source record extracted in previous sections will be translated into an CA Business Service Insight Event.

This section also defines how to handle duplicate Events and how to use the Event singularity mechanism (for further details see Event Singularity (see page 126)).

When the interface mode is set to online, the CA Business Service Insight Event has a unified structure that contains the following fields:

- **Timestamp**: Time of event occurrence.
- **ResourceId**: Resource Id associated with the Event (the Resource that was measured within that event).
- **EventTypeId**: Event type associated with the Event and describes the type of the Event (type of measurement of the Resource, type of ticket action, and so on).
- **DataSourceId** (optional): Any value. Supplies additional filtering criteria for raw data events.
- **Value** (multiple): Value(s) of the Event (measurement result, ticket number, and so on). This field often appears more than once.

The structure of the Translator corresponds with the structure of the Event Type within CA Business Service Insight, and also the database table T_RAW_DATA that stores the event as shown in the following figure:



- **Translator**: Describes how to translate the set of fields it receives into the output Event.
- **TranslatorName**: Name used by the InputFormat to send field sets to this translator. This example uses the name 'Translator1'. Refer to the previous figure for values that could be used for this scenario.
- **TranslatorFields**: Contains a list of TranslatorField elements, each one of which contains the following attributes:
  - **Name**: Field name. In the online interface it must be Timestamp, ResourceId, EventTypeId, ResourceId, or Value.
  - **SourceType**: Specifies the source of the field value. Can be one of the following:
    - **Field**: Value of this field is taken from field in the input format. The SourceName attribute contains the InputFormat field name.
    - **Table**: Value of the field is taken from the Translation table. The SourceName attribute contains the Translation table name that refers to a name which is defined in the next section of TranslationTableCollection. This type is used for values which are chosen to be translated into Resources and Event types in the event.
    - **Lookup**: Value of this field is taken from the Translation table. The SourceName attribute contains the table name. The value to be translated from the LookupValue attribute and not from the input format. It is commonly used when a constant value is required for the purpose of Translations.
    - **Constant**: Value of the field is constant, and its value is in the ConstantValue attribute. When using a constant value, it is necessary to specify the field type by using the Type attribute.
  - **SourceName**: Contains the field name or translation table name.
  - **LookupValue**: Contains the lookup value when SourceType="lookup".
  - **ConstantValue**: Contains the constant value when SourceType=constant. When the field's type is time, the constant value is a formatted string according to the TimeFormat (Set in the General section of the Adapter) or "Now" or "NowUtc", where "Now" is the current time in the Data source locale and "NowUtc" is the current time in UTC.

This section also contains the mapping tables defining the mapping of data source values into CA Business Service Insight Event fields and holds the table definition with the referred data source value which is to be to be translated.

- **LoadingMode**: Default for online interface is "remote" and for offline interface is "standalone".

  Specified loading method of the translation tables as following:

- **Standalone**: Adapter loads the Translation tables locally. There is no connection to the CA Business Service Insight server regarding the Translation. Changes in the Translation tables are stored only in the local file.
- **Remote**: Adapter sends a request to load all the tables from the CA Business Service Insight server. Changes made in the Translation tables are also stored locally.

- **TranslationTable**: Links the event value to the mapping table.
  - **Name**: Name of Translation table used and referred to by the Translator.
  - **DestinationType**: [resource, event_type, contract_party, service, time_zone, value]. Holds the type of the Translation table. In this example, the **Servers** column in the data source file is translated into an CA Business Service Insight Resource. Therefore, the Translation table type is Resource, and it holds translations of values to Resource IDs in CA Business Service Insight.
  - **TranslationField**: Field name to translate from and which is taken from the input format fields. Can contain a maximum of five fields.

Each translation table defined in the configuration file must have a corresponding definition in the CA Business Service Insight user interface.

The XML representation of a sample configuration file is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<AdapterConfiguration>
     <General MajorVersion="4" MinorVersion="0" RunOnce="yes" LogDebugMode="yes"
ConsoleDebugMode="yes" WorkingDirectoryName="output"
RejectedEventsUpperLimit="10000">
     <DataSourceDifferenceFromUTC DefaultOffset="0" TimeFormat="%d/%m/%Y %H:%M">
          <DaylightSaving From="20/04/2001 00:00" To="15/10/2001 00:00" Shift="1"/>
     </DataSourceDifferenceFromUTC>
</General>
     <OblicoreInterface Mode="online">
          <OnlineInterface Port="5555" SecurityLevel="none"/>
     </OblicoreInterface>
     <DataSourceInterface>
          <Files>
               <File DeleteFileAfterProcessing="no" InputFormat="InputFormat1"
NamePattern="servers*.csv" Path=" C:\Program
Files\Oblicore\Adapters\ServersAdapter\data\" TitleExists="yes" SleepTime="60"
Delimiters=","/>
      </Files>
</DataSourceInterface>
<InputFormatCollection>
     <InputFormat InputFormatName="InputFormat1">
          <InputFormatFields>
               <InputFormatField Name="resource" Type="string"/>
                    <InputFormatField Name="timestamp" Type="time"
TimeFormat="%d.%m.%Y %H:%M"/>
                         <InputFormatField Name="memory_util" Type="real"/>
```

```
                        <InputFormatField Name="cpu_util" Type="real"/>
                </InputFormatFields>
     <TranslatorSwitch DefaultTranslator="Translator1"/>
     </InputFormat>
</InputFormatCollection>
<TranslatorCollection>
<Translator TranslatorName="Translator">
         <TranslatorFields>
            <TranslatorField Name="ResourceId" SourceType="table"
SourceName="ResourceTable"/>
            <TranslatorField Name="EventTypeId" SourceType="lookup"
SourceName="EventTable" LookupValue="PerformanceEvent"/>
            <TranslatorField Name="Timestamp" SourceType="field"
SourceName="timestamp"/>
            <TranslatorField Name="Value" SourceType="field"
SourceName="memory_util"/>
            <TranslatorField Name="Value" SourceType="field"
SourceName="cpu_util"/>
         </TranslatorFields>
      </Translator>
   </TranslatorCollection>
   <TranslationTableCollection LoadingMode="remote">
      <TranslationTable Name="ResourceTable" DestinationType="resource">
         <TranslationField>resource</TranslationField>
      </TranslationTable>
      <TranslationTable Name="EventTable" DestinationType="event_type">
         <TranslationField>resource</TranslationField>
      </TranslationTable>
   </TranslationTableCollection>
</AdapterConfiguration>
```

SQL Adapters

SQL Adapters are basically the use of the SQL Adapter generic component (SQL Adapter executable) with the appropriate settings in the configuration file. The SQL Adapter can connect to all data sources that support ODBC and OLEDB. Connection is established via connection string. The appropriate driver should be installed on the server on which the Adapter is installed.

Data sources examples:

- Oracle
- SQL Server
- Access
- Excel
- Text files, CSV files (these could also be connected to via the TEXT adapter, but ODBC connection often provides extra filtering/querying capabilities).

SQL Adapter workflow:

- Open the connection.
- Replace local variables with the last key field values.
- Auto-complete, build the where clauses of the queries and concatenate to the end of the queries.

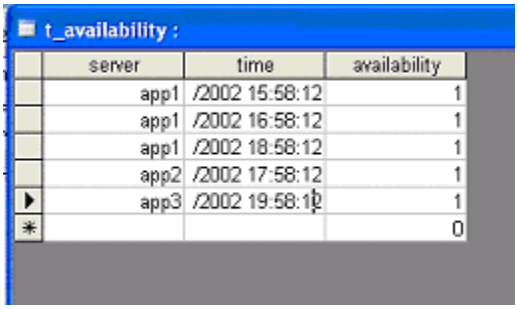
  Execute the query and get recordset.

  For each record in the recordset returned by the query:

  - Find the correct InputFormat.
  - Build the Event record.
  - Translate the record.
  - Update the key fields value in the control file.
- Close the connection.
- Go to sleep.

SQL Adapter Configuration File Example

Given an MS Access (.mdb) database with the following table:


t_availability :

server	time	availability
app1	/2002 15:58:12	1
app1	/2002 16:58:12	1
app1	/2002 18:58:12	1
app2	/2002 17:58:12	1
app3	/2002 19:58:12	1
		0

The only difference in the configuration file in a SQL Adapter configuration file and the file Adapter configuration file is the DatasourceInterface section.

The DatasourceInterface section in a file Adapter stores the Files collection, where the SQL Adapter file has the ConnectionString and QueryCollection.

The main difference between the two configuration files is in the retrieval and parsing method. The rest of the file is the same.

The SQL interface defines the connection to the database and the queries used to retrieve the data.

Details are as follows:

Note: The section is defined based on the above data source database.

Connection string element

ConnectionString: Connection string for accessing the source DataBase.

The ConnectionString can be defined in the DataSourceInterface element and/or in the Query elements. The ConnectionString definition in the DataSourceInterface element is the default and is used only in a query without a ConnectionString definition. This allows the Adapter to connect to several databases where each query can have its own connection string. For more details on the connection string mechanism, refer to the following section.

Configuration File Query Collection Section

Query: Specifies the query attributes.

- **QueryName**: Name for the query.
- **InputFormat**: InputFormat associated with this query. The Adapter uses the InputFormat to extract the data from the source.

  Note: The order of the InputFormat fields has to correspond to the order of the query selected fields.
- **SleepTime**: Time in seconds during which the Adapter sits idle to wait for new data to arrive.
- **SelectStatement**: Contains selected statement to be executed on the source database.

  Note: You must enter the query key fields as the first selected values in the statement.

  - **AutoCompleteQuery**: When set to "yes", the Adapter automatically concatenates a where statement to the specified query as follows:
    - Creating a where statement that retrieves only new values based on the key fields.
    - Ordering the results statement based on the key fields.
- **QueryKeyFields**: Defines the fields to start the next data retrieval.
  - **KeyField**:
    - **Name**: Name of the field, taken from the fields of the query.
    - **Sort**: Data sorting order (ascending/ descending).
    - **ValueLeftWrapper**: Concatenates characters before the value of the field. The default is ' (apostrophe).

- **ValueRightWrapper**: Concatenates characters after the value of the field. The default is ' (apostrophe).
- **Critical**: Stops executing other queries if this particular query fails.
- **SleepTime**: Sleep time between required operations. The default is ' (apostrophe).

Note: 'Date' fields often require using special characters to enclose the date itself. The characters needed to identify the field as a date field depend on the data source. The # character, as shown in the figure at the beginning of the section, can be used to wrap the value field in Excel. Other data source however, may require different methods. Refer to Adapter Configuration Specifications (see page 311) for more information.

- **SelectInitialValues**: A select statement that provides the initial values for the key fields for the first where statement (When the control file is empty).

  Example of a query (Excel-based ODBC) that is built with AutoCompleteQuery="yes":

  SELECT INC_CLOSE_DATE, INCIDENT_REF, Severity, `Resolve Time`, `Date Logged`, `Date Resolved` FROM `AllCalls$` WHERE INC_CLOSE_DATE>CDate('7/03/2005 13:06:21') order by INC_CLOSE_DATE

  This select statement must be executable on the target DB on which the query is running. This may differ between sources and the ODBC drivers used to connect to them. For example, in Oracle you could select the values from the special 'dual' table (select 'aaa', '1-jan-1970' from dual) , but in Excel you can just select the values directly without a table. (select 'aaa')

Following is a complete configuration file in XML layout:

```
<?xml version="1.0" encoding="utf-8"?>
<AdapterConfiguration>
  <General MajorVersion="3" MinorVersion="0" RunOnce="yes" LogDebugMode="yes"
ConsoleDebugMode="yes" WorkingDirectoryName="d:\Oblicore\Training
Kit\Exercises\Adapters\SQL Adapters\Ex1\Solution">
      <DataSourceDifferenceFromUTC DefaultOffset="1"
TimeFormat="%Y/%m/%d-%H:%M:%S"/>
   </General>
   <OblicoreInterface Mode="online">
      <OnlineInterface Port="2000" SecurityLevel="none"/>
   </OblicoreInterface>
   <DataSourceInterface>
      <ConnectionString>

          Driver={Microsoft Access Driver (*.mdb)};Dbq=d:\Oblicore\Training
Kit\Exercises\Adapters\SQL Adapters\Ex1\db1.mdb;

      </ConnectionString>
      <QueryCollection>
         <Query QueryName="Query" InputFormat="InputFormat" SleepTime="5">
            <SelectStatement AutoCompleteQuery="yes">
```

```
                  select time,server,availability from t_availability

            </SelectStatement>
              <QueryKeyFields>
                 <KeyField Name="time" Sort="asc" ValueLeftWrapper="#"
ValueRightWrapper="#"/>
                 <KeyField Name="server" Sort="asc"/>
                 <SelectInitialValues>

                       select 'AAA','1/1/1970'

                 </SelectInitialValues>
              </QueryKeyFields>
           </Query>
        </QueryCollection>
     </DataSourceInterface>
     <InputFormatCollection>
        <InputFormat InputFormatName="InputFormat">
           <InputFormatFields>
              <InputFormatField Name="timestamp" Type="time" TimeFormat="%m/%d/%Y
%I:%M:%S %p"/>
              <InputFormatField Name="server" Type="string"/>
              <InputFormatField Name="status" Type="integer"/>
           </InputFormatFields>
           <TranslatorSwitch DefaultTranslator="Translator"/>
        </InputFormat>
     </InputFormatCollection>
     <TranslatorCollection>
        <Translator TranslatorName="Translator">
           <TranslatorFields>
              <TranslatorField Name="ResourceId" SourceType="table"
SourceName="ResourceTable"/>
              <TranslatorField Name="EventTypeId" SourceType="constant"
ConstantValue="1500"/>
              <TranslatorField Name="Timestamp" SourceType="field"
SourceName="timestamp"/>
              <TranslatorField Name="Value" SourceType="field" SourceName="status"/>
           </TranslatorFields>
        </Translator>
     </TranslatorCollection>
     <TranslationTableCollection LoadingMode="remote">
        <TranslationTable Name="ResourceTable" DestinationType="resource">
           <TranslationField>server</TranslationField>
        </TranslationTable>
     </TranslationTableCollection>
</AdapterConfiguration>
```

SQL Adapter Connection String

The SQL Adapter connection string mechanism is designed to provide the following capabilities:

- Work with several connection strings in the same Adapter.
- Work with a connection string template, where the file name can be changed without changing the configuration file.
- Delete the file data source once the Adapter has finished reading it.
- Move the file data source to another location once the Adapter has finished reading it.

In addition to the simple definition of the connection string as a string in the ConnectionString element, the connection string can be defined by segments, where each segment holds the specific values that are concatenated to the connection string. This allows the Adapter to build the connection string dynamically.

There are two segment types. One is text and contains text that is concatenated to the connection string as is. The second is a file and contains a file name with or without wildcards. The file segment can appear only once. It contains other attributes that define what must be done with the read file.

The ConnectionString can be defined in the QueryCollection element and/or in the Query elements. The ConnectionString definition in the QueryCollection element is the default and is used only in a query without an explicit ConnectionString definition.

Elements & Attributes

ConnectionString Element

This element can contain segment child elements. If it contains at least one segment element, the connection string is concatenated with that. Otherwise, the connection string is taken from the ConnectionString element text (as in the current version).

Segment Element

This element contains one mandatory attribute called *Type*, which has two legal values; text and file. When the Type="file", the Segment element must contain at least one File element. Each File element is considered a different query.

File Element

This element contains attributes that define which file(s) should be used in the connection string and what must be done with the file when the Adapter has finished reading it.

- **Path**: Defines the file Path (directory).
- **NamePattern**: Defines the file name with the given path. It may contain wildcards.
- **UsePath**: Legal values: yes/no. The default is "yes". If set to :yes"', the file path is concatenated to the connection string.
- **UseFileName**: Legal values: yes/no. The default is "yes". If it is set to "yes", the file name is concatenated to the connection string (as needed for Excel files).
- **UpdateSchemaFile**: Legal values: yes/no. The default is "no". If it is set to "yes", the Adapter updates the schema.ini file with the current file name.

  Note: Use this attribute only when you want the Adapter to change the schema.ini file.

Internal variables

Two additional internal variables have been added and can be used in the SelectStatement and SelectInitialValues elements. These are:

- :file_name: Replaced by the current file name and extension.
- *:file_name_no_ext*: Replaced by the current file name without the extension.

Examples

Example 1: Simple connection string for Oracle:

```
<ConnectionString> Provider=msdasql;dsn=0; uid=0; pwd=0 </ConnectionString>
```

or

```
<ConnectionString>
   <Segement Type="text" Text="Provider=msdasql;"/>
```

```
   <Segement Type="text" Text="dsn=0; "/>
   <Segement Type="text" Text="uid=0; "/>
   <Segement Type="text" Text="pwd=0; "/>
</ConnectionString>
```

Example 2: Simple connection string for Excel with a single file:

```
<ConnectionString>Driver={Microsoft Excel Driver (*.xls)}; DriverId=790;
                                Dbq=d:\Oblicore\Availabilty_2003_01.XLS
</ConnectionString>
```

or

```
<ConnectionString>
   <Segement Type="text" Text=" Driver={Microsoft Excel Driver (*.xls)};"/>
   <Segement Type="text" Text=" DriverId=790;"/>
   <Segement Type="text" Text=" Dbq="/>
   <Segement Type="File">
        <File Path="d:\Oblicore " NamePattern="Availabilty_2003_01.XLS">
   </Segement>
</ConnectionString>
```

Example 3: Simple connection string for use with several Excel files:

```
<ConnectionString>
   <Segement Type="text" Text=" Driver={Microsoft Excel Driver (*.xls)};"/>
   <Segement Type="text" Text=" DriverId=790;"/>
   <Segement Type="text" Text=" Dbq="/>
   <Segement Type="File">
        <File Path="d:\Oblicore ",NamePattern="Availabilty_*.XLS"/>
   </Segement>
</ConnectionString>
```

Example 4: Using a standard ODBC DSN entry:

Using a standard ODBC DSN entry you can connect to any source that has a DSN entry created in the ODBC manager on the application server. The standard ODBC DSN entry can be found in the 'Administrative Tools' section of the server.

```
<ConnectionString>dsn=SampleDataSource;usr=scott;pwd=tiger;</ConnectionString>
```

Reading Records from a File Data Source

As long as there is an ODBC interface to the data source, it is possible to configure an SQL Adapter to query files. To configure the Adapter to read from multiple files, it is necessary to use the Segment elements in the ConnectionString element. For an example, see the previous section describing the connection string.

The way in which the SQL Adapter works with the files is as follows:

1. In each query, the Adapter attempts to read from the file until it cannot retrieve any more records.
2. The adapter then moves to the next file and attempts to read it.
3. When there are no more files, the Adapter goes to sleep for this query in accordance with the SleepTime attribute.

Schema.ini File

When a text ODBC driver is employed, the format of the text file is determined by the schema information file (schema.ini). This schema information file should be placed in the same directory as the text data source.

Schema.ini files are built out of entries, with each entry describing the structure of a single text file. The first line in each entry is the name of the text source file, enclosed in square brackets.

When the Adapter is working with files that were defined using wildcards, the file name changes constantly. Since the name in the schema.ini file cannot contain wildcards, the Adapter must change the schema.ini file before connecting to the database.

You therefore need to add an indicator line before the file entry line. This indicator line contains the name pattern from the connection string element in the Adapter configuration file, enclosed in square brackets. The Adapter replaces the next line with the new file name enclosed in square brackets.

Note: The schema.ini file can contain several entries. It is your responsibility to add the line(s) in the correct place.

Schema.ini Example

```
[sqltes*.txt]
[sqltest2.txt]
ColNameHeader = False
CharacterSet = ANSI
Format = TabDelimited
Col1=id Char Width 30
Col2=idname Char Width 30
----------------------------------------
[report_200*.txt]
```

```
[report_2003_09.txt]
ColNameHeader = False
CharacterSet = ANSI
Format = TabDelimited
Col1=date Char Width 30
Col2=service Char Width 30
Col3=responseTime Char Width 30
---------------------------------------
```

DataSourceControl File

For each query, the data source control file contains the file name pattern and the name of the current input file to continue with the same file when the Adapter restarts.

Following is an example of the control file in its XML format:

```
<AdapterControl Save="end" LastSaveTime="2005/03/23 09:16:15">
   <Data>
      <QueryCollection>
         <Query QueryName="TroubleTickets (on D:\Data\Incidents*.xls)">
            <KeyField Name="INC_CLOSE_DATE">
               <LastValue>7/03/2005 13:06:21</LastValue>
            </KeyField>
            <LastFileName>IncidentsMarch.xls</LastFileName>
         </Query>
      </QueryCollection>
   </Data>
```

Retaining input files

When the Adapter has finished reading the current file it searches for the next one. The next file read is the first file that fits the name pattern and whose name is greater (in lexicographical order) than the previous file name. The Adapter does not return to previous files even if new records are added to them.

The Adapter uses the InitialFileName attribute only when these two attributes are equal to "no", and the control file does not contain the last file name.

Queries checking

The Adapter checks the connection string and the query only if the defined file exists. If it is defined using wildcards, the Adapter checks it only against the first file.

Errors can occur when the Adapter tries to read from a new file. In this case, the Adapter stops immediately if the critical attribute is equal to "yes". If it is equal to "no", the Adapter does not continue to perform this query, but continues with the other queries. Alternatively, the Adapter leaves the current file and moves to the next file.

Adapter Internal Variables

There are two internal variables which can be used in the configuration file to refer to the current context of the file name. These are :file_name and :file_name_no_ext , referring to the current file name and the current file name minus the file extension, respectively.

These variables can be used in the SelectStatement element, SelectInitialValues element and in the QueryKeyField\Function attribute.

The Adapter replaces the variable with the file name in the queries.

For example:

- select date, service, value from ":filename"
- select id and name from :file_name_no_ext

Create an Adapter Using the CA Business Service Insight User Interface

Each Adapter configured to run within the CA Business Service Insight environment must be registered within the user interface in addition to being defined in the registry. The reason behind this step is mainly to establish the settings of the Adapter Listener side so that it is ready to receive events from the Adapter. During this step the all of the Adapter settings, such as, Translation tables and Event types are defined.

Follow these steps:

1. Create the Adapter.
2. Choose Resources/Adapters.
3. Check the existing Adapters in the list to see that none are defined on the same port as your Adapter, that is, the same port that is defined in the configuration file of the Adapter in the directory OblicoreInterface\OnlineInterface\Port.
4. Click Add new, and then select the method that you want to use to create the Adapter. There are a number of options:
    a. **Create manually**: Sets up the Adapter Listener to connect to the Adapter you have defined (or will define) manually.
    b. **Using the Wizard**: Allows creating the adapter using a screen-by-screen wizard interface. See the next section for details on this.
    c. **From a template**
    d. **From an existing configuration file**: Allows uploading a pre-configured adapter template which automatically fills in the fields of the adapter wizard with the options set in the configuration file specified.
    e. The resulting screen varies per chosen option.
5. Fill in the fields:

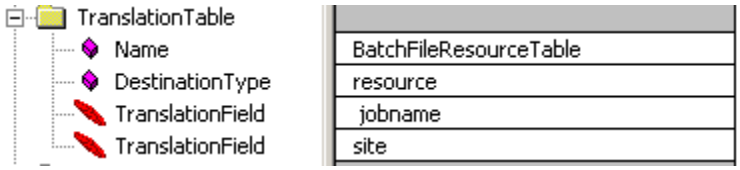
    Network address-enter the IP address of the Adapter. localhost in case it is local to the application server, otherwise enter the defined port.
6. Click Save.

To create the necessary translation tables:

Note: These steps should be performed for each translation table defined in the configuration file:

1. On the Design menu click Translation, Translation Tables and click the 'Add New' button.
2. All the translation table settings should correspond to the equivalent table definition in the Adapter configuration file:
    – *Name*: Should match the Name attribute in the configuration file Translation table, Name.

- Source fields: Should have all the fields from the TranslationField of the translation table. Add all fields. There may be a combination of two or more fields that make up the translation value. The following is an example of how this might look:



3. Destination type: Should again correspond with the DestinationType attribute of the translation table definition in the configuration file. (resource, event_type, and so on).
4. Registered Adapters: Add the Adapters that are to use this translation table. More then one Adapter can use a single translation table.
5. Click Save.
6. Import the Event type fields definition for each event type.

   To be able to import the fields definition from a specific Adapter configuration file, this Adapter should have run and connected to CA Business Service Insight at least once. When the Adapter connects to CA Business Service Insight, it sends the configuration file to CA Business Service Insight which allows the system to use the fields definition from it.

   Note: Alternatively, you can specify the field definitions manually to the event type.
7. Activate the Adapter:
   a. On the Design menu, click Data Acquisition, Adapters.
   b. Click the Start button for the Adapter.

The following table contains the different statuses of the Adapters:

Status	Description
Inactive	Initial state. Adapter is inactive and has not yet been started.
Listener Inactive	Adapter listener (Dispatcher) Service is not started.
Starting	Adapter is starting.
Started	Adapter is started.
Stopping	Stopping.
Pausing	Pausing.
Paused	Paused.
Not running	Adapter is not running on Adapter's computer.
Error	Error in Adapter's configuration file; Adapter cannot start.

Connection error	Error connecting Adapter (wrong host/port), or before Adapter has run for first time and no signal was detected. Status when running Adapter for first time.
Blocked	Maximum number of rejected Events has been reached.

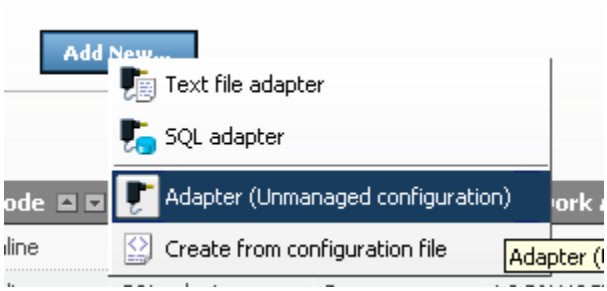
Create an Adapter using the Adapter Wizard

The Adapter wizard is a newer feature of the CA Business Service Insight User Interface and provides guidance in creating a new Adapter, using a more intuitive interface than the XML Editor. The wizard guides you through a series of tabs containing all the information required to create an Adapter, and at the end allows you to download a copy of the compiled XML configuration file. There are, however, some limitations to what the wizard can do.

It currently does not allow you to:

- Refer to the same input structure (Input Format) from different data source interfaces
- Refer to the same output structure (Translator) from different inputs
- Configure an Input Format Switch and use it to decide which Input Format to use from the data source interface
- Provide a definition of a Data Source ID field in the output structure
- Provide a definition of more than one file in the connection string to query text or Excel files with the same query
- Use UTC or UtcNow time constraints
- Specify values that start or end with a 'space'

When creating a new Adapter via the wizard there are four options to choose from, as shown in the following figure:



The first two options allow you to create either a Text File Adapter or SQL Adapter using the Wizard interface. The next option Adapter (Unmanaged Configuration) is the option to choose when adding a preconfigured Adapter that you have built using the XML Editor. This option does not allow you to further edit the configuration for this Adapter using the wizard at a later time. The final option Create from Configuration File allows you to upload a pre-configured Adapter configuration and have the system import it into the wizard interface for further editing. This option requires that none of the wizard limitations mentioned above is implemented in that particular configuration.

Beyond this point, the configuration options of the Adapter wizard provide the same functionality described for the manual Adapter configuration. Its purpose is to provide a simpler and friendlier interface for editing the configuration settings. Since the same functions and principles as in the manual alternative apply refer to the relevant sections for details.

Execute and Test the Adapter

The setting of the Adapter configuration file cannot usually be completed in a single cycle. It may need a number of iterations during which the Adapter is executed, and the results are checked in order to ensure that the Adapter configuration is correct.

Following are some of the more common issues that need to be corrected:

- Connection problems (either between data source and Adapter or Adapter and Listener)
- Mistakes in the configuration file, such as:
  - Wrong structure
  - Wrong use of attributes
  - Wrong case used (Adapter are case sensitive - 'Yes' and 'yes' are different, and so on)
  - Wrong value assignment
- Data manipulation mistakes, such as, output Event structure, wrong event values, mistake in queries.

Follow these steps:

1. Set the Adapter to RunOnce = “yes” and LogDebugMode = “yes”, as well as setting the RejectedEventsUpperLimit to a reasonable number (see Adapter Running Modes (see page 85)).

   The following figure shows the configuration settings required for testing.
2. It is also possible to use the offline mode for the purposes of the configuration file settings.
3. Once the configuration file has loaded successfully, change the setting back to online (see Adapter's running modes).
4. Each iteration includes the following steps:
   a. Update/fix the Adapter configuration file.
   b. Delete all of the Adapter's output files.
   c. Run the Adapter by double-clicking the shortcut to the *Adapter* executable or the *.bat* file that was created.
   d. Open the Adapter's log file using the Log browser (Log Browser.exe utility placed in CA Business Service Insight Utilities) and check that there are no error messages.
5. The first step is to get a correct configuration file and to reach a state where the Adapter loads the configuration file successfully. Repeat this step until you successfully connect to CA Business Service Insight and to the data source and have rejected events and requests for translation.
6. To complete this stage check the following:

- There are no error messages in the Adapter's log file.
- Adapter connects to CA Business Service Insight and the data source successfully.
- Adapter sent requests for translations and rejected events.

You should expect to see the letter "R" appear on the console for every record the Adapter rejects. Remember that rejected events are expected until all of the necessary translations have been completed.

7. Check that the rejectedEvents file contains records and is not empty.

8. Login to CA Business Service Insight and go to the *Translation entries* page and search for pending Translation entries from your Adapter. You should expect to have several entries, one for each Translation request the Adapter has sent.

   WARNING: Deleting the Adapter's output files is very risky. It should be done only at this stage for testing purposes. For example, when deleting the control file, the Adapter loses track of the files it has already read and may therefore lose data or read files again. The only file that can be deleted during operational mode without functional consequences is the log file.

To use the log browser to view the error messages:

If there is an error message, double-click the message and read it. This is usually caused by an error in the configuration file.

Resource and Event Translations

In the previous step, there were a number of Rejected Events created during the execution of the Adapter. These Rejected Events were stored in a RejectedEvents.txt file, but were also stored as Pending Translation entries in the CA Business Service Insight Database. The next step in the process of loading raw data into the system is to provide a translation of what has been measured so that CA Business Service Insight can use this data as required.

Raw Data events could be rejected either due to the Event Type OR the Resource not being defined in the system. The pending events are defined by what type of Translation Table they came from. The most common examples are having the Event Types coming from one Translation table, and the resources coming from another Translation table.

When a new Resource is encountered by the Adapter (say for example, that a new server is added to the network monitoring tool and shows up as a new entry in the data source from this monitoring tool) it can be added to the resource model of the system. There are two steps you need to take for this new resource to become reportable in CA Business Service Insight.

You first need to create the resource as an CA Business Service Insight entity (a resource) and then a translation needs to be entered. This provides the link between the string representation found in the data-source, and the entity defined as a Resource in CA Business Service Insight. This 2-step process can be performed in a single action via the user interface in a process known as 'Add and Translate' whereby the new resource and the necessary Translation entry may be created in a single procedure. When adding and translating, multiple entries may be selected provided they have the same allocation settings. Performing Translation is the process of creating the match between the data source value and the CA Business Service Insight Entity. When creating Translations, an entry is added to the Translation table with the matched values. Then, on subsequent queries to the data-source, the adapter will automatically know how to handle this new value.

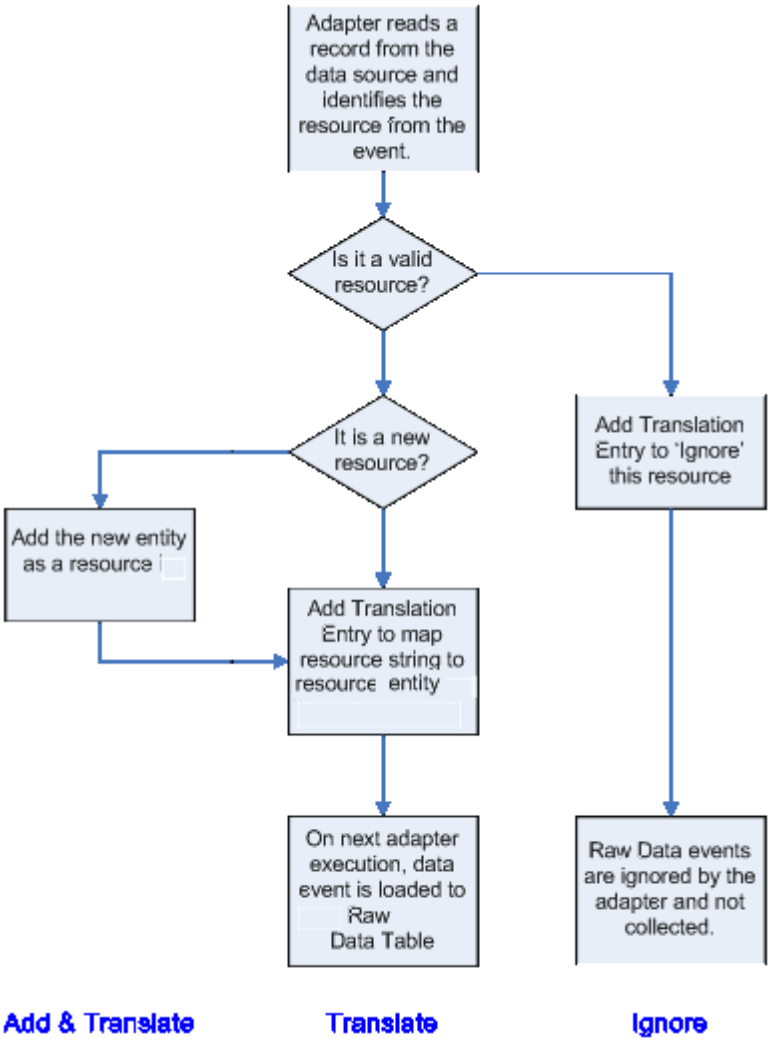
At this stage, the Adapter has already run and sent requests for Translation for each value in the Translation fields. The events associated with these values were rejected and are waiting to be sent to CA Business Service Insight once the Translation has been completed. Translations can be done manually or automatically using a Translation script.

The following Translation actions are possible for pending Translation entries:

Translate: Allows creation of an entry in the Translation table, making the match between a data source value and the relevant CA Business Service Insight Entity. The CA Business Service Insight Entity to which Translations are made must already exist. (A good example of this may be in the case of a misspelled resource from a data source. There may be different names given in the data-source which are actually referring to a single logical entity. That is, Server001 and SERVER001. CA Business Service Insight resources are case-sensitive.)

- **Add and Translate**: Allows creation of a new Entity in CA Business Service Insight *and* adds a Translation entry for that Entity in the Translation table at the same time. This is the most common action performed on pending Translation entries because the Translation mechanism is used to build the infrastructure in CA Business Service Insight.
- **Ignore**: When ignoring an entry, all associated events are ignored and not sent to the CA Business Service Insight raw data table. The ignored events will be lost. For example, if the data source contains information on all the servers in a data center, but only the application servers' data is required for service level calculation, then all servers will reach the Adapter for Translation, but only the application servers will be Translated. All others are ignored since CA Business Service Insight ensures that the unnecessary event is ignored as well. An ignored entry can be translated at a later stage if required, but only captures this data from that point forwards.
- **Delete**: Translation entry is removed and the associated rejected Event is also deleted. If the same value is later sent again by the data source, a new pending entry is created.

The following flowchart summarizes the cases for using these options:


Adapter reads a record from the data source and identifies the resource from the event.
Is it a valid resource?
It is a new resource?
Add the new entity as a resource
Add Translation Entry to map resource string to resource entity
On next adapter execution, data event is loaded to Raw Data Table
Add Translation Entry to 'Ignore' this resource
Raw Data events are ignored by the adapter and not collected.
Add & Translate
Translate
Ignore

Manual Translations

Manual translations are required when the entity already exists within CA Business Service Insight. This may occur in a number of situations. For example, when a translation script to automatically create the resources from an external source has been run (but the translations could not be automated). Or when a data source has dubious entries and a resource has been defined twice in different ways (that is, Server001p and server001P). It could even be due to resources that were created manually.

To run a manual translation when the resource already exists:

1. On the Design menu, click Translations, Translation entries.
2. By default, all pending entries are displayed.
3. Select the pending translation entry you want to translate by selecting the check box next to it.
4. Click Translate.
5. Choose the relevant entity (Resource, Event type, and so on) from the list. (If no items are shown in the list you may need to change the default search criteria in the box.)
6. Select the Resource/Event Type from the displayed list of entities by clicking on the line containing the item. It stays highlighted when selected.
7. Click Translate. The translation entry is now stored in the system.

To run a manual translation when the resource does NOT yet exist:

1. On the Design menu, click Translation, Translation entries.
2. By default, all pending entries are displayed.
3. Select the pending translation entry you want to turn into an CA Business Service Insight resource and translate by selecting the check box next to it.
4. Click Add & Translate.
5. Ensure that the resource name is specified as you wish. You can also customize the *Display Name* field to change the way the resource appears on a report. If this resource should be managed as part of a 'Change Set', the specific change set should be specified here too.
6. The Effective Date of the resource should be configured as the date at which this resource should be reportable from in the system. Note that the resource will NOT show up in reports earlier than the date specified here.
7. Click the Details tab and select the following options for the resource association: Effective (true/false), Resource Type, Resource Group membership, Service and Contract association.
8. Click Save & Translate. The resource is added to the resource service catalog and the translation entry is also now be stored in the system. Once the Pending Entries have all been dealt with you can check that the data loads into the system:
9. Browse to the resources in CA Business Service Insight and check that the new resource has been created.
10. Run the Adapter again.
11. Check the rejected events file is empty, content and size.
12. Check the send events file is empty, content and size.
13. Check with the raw data tool that the Events that have been added to the raw data.

Infrastructure Setup

Setting up the infrastructure includes the definition of the Data Modeling entities as identified during the Design phase.

This stage does not include the definition of all infrastructure Entities, and is complete when the Adapter configuration has been completed.

During this stage the following entities are fed into the CA Business Service Insight system:

- Event types
- Resource types
- Resources & Resource allocations
- Resource Groups

Note: Once the Adapter has been executed successfully, you can use the automatic import feature when defining the Event type fields.

Automatic Translations with Translation Scripts

Automatic Translation is the automation of the creation and translation processes and infrastructure, based on an external data source using scripts that run and perform the Translation actions.

Automatic Translation is done through translation scripts. Translation scripts speed up the process of mapping new IT and business resources into CA Business Service Insight. The translation script automatically identifies when a new translation entry is received and translates resources, allowing for quick and efficient resource mapping. The automation supports the interface to CMDBs, enabling the system to identify resources based on their configured definition. Automatic translation has a number of benefits, including easing the maintenance of translations and preventing errors. Translation scripts can be used to create new resources and allocate changes.

Additionally, translation scripts can be used to:

- Translate entries to existing CA Business Service Insight objects.
- Add new CA Business Service Insight objects and translate to them based on existing translation entries.
- Create objects and translate entries on the basis of tables outside CA Business Service Insight, for example, resource tables from another external CMS.
- Check whether an object exists.
- Create resources and perform resource allocations, such as, resource types, resource groups, contract parties, and service components.
- Allocated/Deallocate resources to Change Sets.

Since the translation process is also supported as a manual procedure in the user interface, a decision on which translation process to choose is required. When doing so, consider the following pros and cons concerning automatic translation:

- Increased project complexity-translation scripts require appropriate skills and script development efforts.
- Added development time as well as Quality Assurance time for testing.
- Need not be implemented in cases when the translation process is merely a one-time initial effort.
- Can be scheduled for implementation as part of a second phase approach.
- Reduced ongoing maintenance.
- Reduced human translation errors.
- For further information on creating and executing Translation Scripts please refer to the *SDK* guide in the CMDB Integration chapter.

Adapter Feature Advanced Topics

The following topics address advanced topics for the adapter feature.

Event Singularity

Event singularity is an Adapter mechanism that provides a process which prevents insertion of duplicate raw data into the T_Raw_Data table.

Without Event singularity, when the Adapter runs against a data source and loads Events into the database, no validation is performed for an identical event. The Event Singularity feature resolves this by providing the ability to specify whether or not to check Events for their singularity before insertion and what to do if the situation should occur. This verification process may, however, adversely impact the performance of the Adapter.

The solution enables the user to define a key, which can be based on the fields of the Event. Such a key represents the Event's uniqueness, which means that if two Events have same key, they are identical Events.

It is also possible to specify what action is taken in the event that a duplicate key is already found in the database. These actions are discussed below.

Note: The key can be defined as a combination of several fields from the Translator.

Adapter Configuration File with Event Singularity

TranslatorCollection/Translator/OnDuplication

This field determines what must be done if the Event already exists in the database.

Possible values are:

- **Add**-inserts (additional) Event(s) into the database.
- **Update**-causes the deletion (in some cases) of the existing Event and the insertion of a new one after validating that the event has changed.
- **updateAlways**-causes the deletion (in some cases) of the existing Event and the insertion of a new one without checking for changes.
- **Ignore**-does not insert a new event into the database.

The default value is Add.

Ignore versus Add

There may be minor performance decreases while running the Adapter in the Ignore mode. However, in the Add mode, the system triggers recalculation in every case of a duplicated event.

Update versus updateAlways

Using the Update option decreases the performance of the Adapter, while updateAlways decreases the performance of the calculation while triggering recalculation for every case.

TranslatorCollection > Translator > TranslatorFields > TranslatorField > IsKey

This attribute determines whether the field it belongs to should contribute its value towards the unique key for the raw data. It is included if value = "yes" and not included if value = "no". The default value (if no value is given) is "yes" for standard fields (ResourceId, EventTypeId and Timestamp) and "no" for all others. The key should be chosen carefully to ensure that the raw data maintains the required integrity and ensures the calculations are fully accurate.

Adapter Listener Behavior with Event Singularity

When receiving a new Event from the Adapter, the Listener checks the value of the OnDuplication field. When the value is "add", the regular insertion process is performed. When the value is not "add", the Listener checks for the existence of an Event with the same UniqueKey and the same reader ID in the database. If the database already contains an Event as described, the new Event is not be inserted into the database when the OnDuplication value is "ignore".

When the OnDuplication value is "update", a check for changes in the Event is performed. If all fields are identical, the new Event is not be inserted into the database.

When the OnDuplication value is "updateAlways", the previous check is skipped and an update occurs regardless.

In both update and updateAlways modes, the following steps are taken:

- If ResourceId, EventTypeId, and Timestamp were changed, recalculation is done to all the Metrics that use the old event formula.
- The event is updated with the appropriate values.

Aggregate Transaction Data

Transaction data is often collected in order to match it against thresholds or to be able to calculate the eventual periodic percentages of success. For example, every five minutes a virtual transaction is performed against a system and the result (response time in milliseconds), is stored as seen below:

```
[…]
1/1/2004 00:00 432
1/1/2004 00:05 560
1/1/2004 00:10 329
1/1/2004 00:15 250
1/1/2004 00:20 275
1/1/2004 00:25 2860
1/1/2004 00:30 140
[…]
```

In other situations, instead of using virtual transactions, there may be access to actual transactions taking place in a system. In these cases, hundreds or even thousands of transactions may be performed hourly.

In both of the above cases, loading such a volume of information into CA Business Service Insight should be avoided, if at all possible.

Aggregating the data by time periods is the best way to reduce the amounts of data. When the threshold against which success is measured is fixed, aggregation can be performed by allowing the Adapter to count the number of transactions within the aggregated period that were successful. For example, if the success threshold in the previous example is set at 500 milliseconds, only five out of seven transactions were successful within the displayed period. The problem with this approach is the fixed threshold: What if one should, later on, wish to change the threshold? In such a situation, raw data must be reread and tested by the Adapter against the new threshold.

Therefore, optimally the Adapter should aggregate transaction data in a flexible manner without losing significant data.

A limited solution is to allow the Adapter to test the transactions against several thresholds. There are two ways of doing this:

- One event with several tests-Event Type = {TotalTransactions, TransBelow250, TransBelow500, TransBelow750, […]}
- Several events with one test-Event Type = {TotalTransactions, Threshold, TransBelowThreshold}.

Both options suffer from the same problem-thresholds can be changed in the future only within a small set of predefined values.

Recommended Solution

Assumption-all potential thresholds are a multiple of a certain number. For example, all thresholds (in milliseconds) are a multiple of 250, so 0, 250, 500, 1750 and 3000 milliseconds are all potential thresholds.

Based on this assumption the suggested solution is to aggregate transactions by rounding all values to the common multiple, and counting how many transactions fall into each rounded value. Event Type = {RangeFrom, RangeTo, TransactionCount}

For example, the following Events will be generated in order to aggregate the data displayed above, where the common multiple is 250 milliseconds:

```
{1,     250,    2}
{251,   500,    3}
{501,   750,    1}
{751,  1000,   1}
```

Comments:

The timestamp of those events would be the same.(For example all aggregated events may occur at 1/1/2007 00:00., and there may be another set of events for the next time sample at 1/1/2007 01:00 assuming hourly aggregation)

RangeTo is calculated by rounding a transaction to the common multiple (see next).

RangeFrom is RangeTo minus the multiple, plus 1. It is specified for clarity reasons only, and you may skip it.

For example-aggregating by the hour would appear similar to the following (replace MULTIPLE by the multiple value):

```
select  trunc (time_stamp, 'hh') "TimeStamp",
        round (response_time/MULTIPLE, 0)*MULTIPLE-MULTIPLE+1 "RangeFrom",
        round (response_time/MULTIPLE, 0)*MULTIPLE "RangeTo",
        count (*) "TransactionCount"
from    t_log
group by trunc (time_stamp, 'hh'),
        round (response_time/MULTIPLE, 0)*MULTIPLE
```

In the Business Logic, the following condition may be applied to the Events:

```
If eventDetails("RangeTo")<=Threshold Then
        SuccessfulTransactions=SuccessfulTransactions+eventDetails("TransactionCo
unt")
End If
```

Some conclusive insights:

- Since transactions tend to distribute normally, the number of aggregative events should be relatively low.
- Selecting higher common multiples will derive less aggregative events.
- Aggregative event volume should always be less or equal to the raw data volume.

Run an Adapter Behind a Firewall

In order to be able to run an Adapter behind a firewall the following solution is suggested:

Open two ports; the first port is the port assigned to the Adapter in CA Business Service Insight; the second port, which is optional but recommended, is the log server port. The log server port is defined by default as port 4040. Opening the log server port enables the Adapter to report to the CA Business Service Insight log and also allows it to generate alerts. This is optional because the Adapter always reports to a local log file.

For both ports the protocol in use is TCP/IP.

Load Data from Several Directories

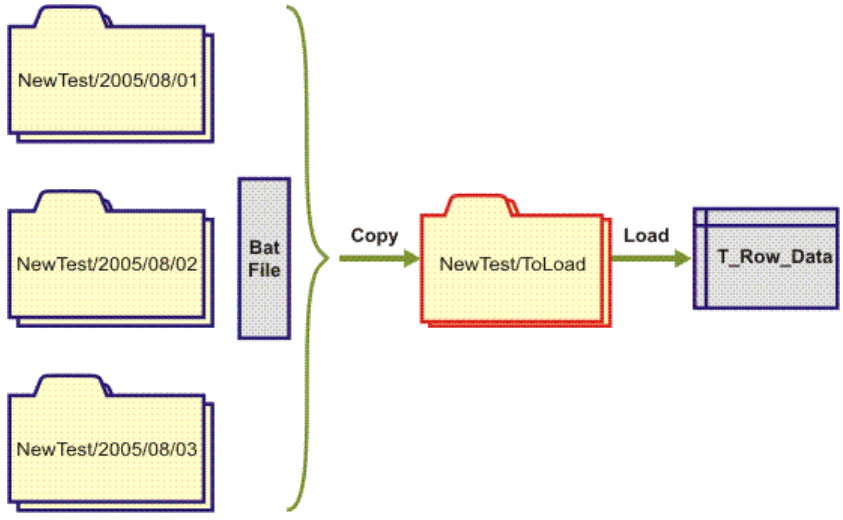
This section describes a solution that can be implemented if the data source files (input to an CA Business Service Insight Adapter) are located in different directories each day, or for all set periods of time (according to a specific naming convention).

For example, given the directory structure c:\NewData\YYYY\MM\DD. In this instance, for each day there is a new directory in which the relevant files for that day are located.

The CA Business Service Insight Adapter needs to browse through the newest directory and load the new files.

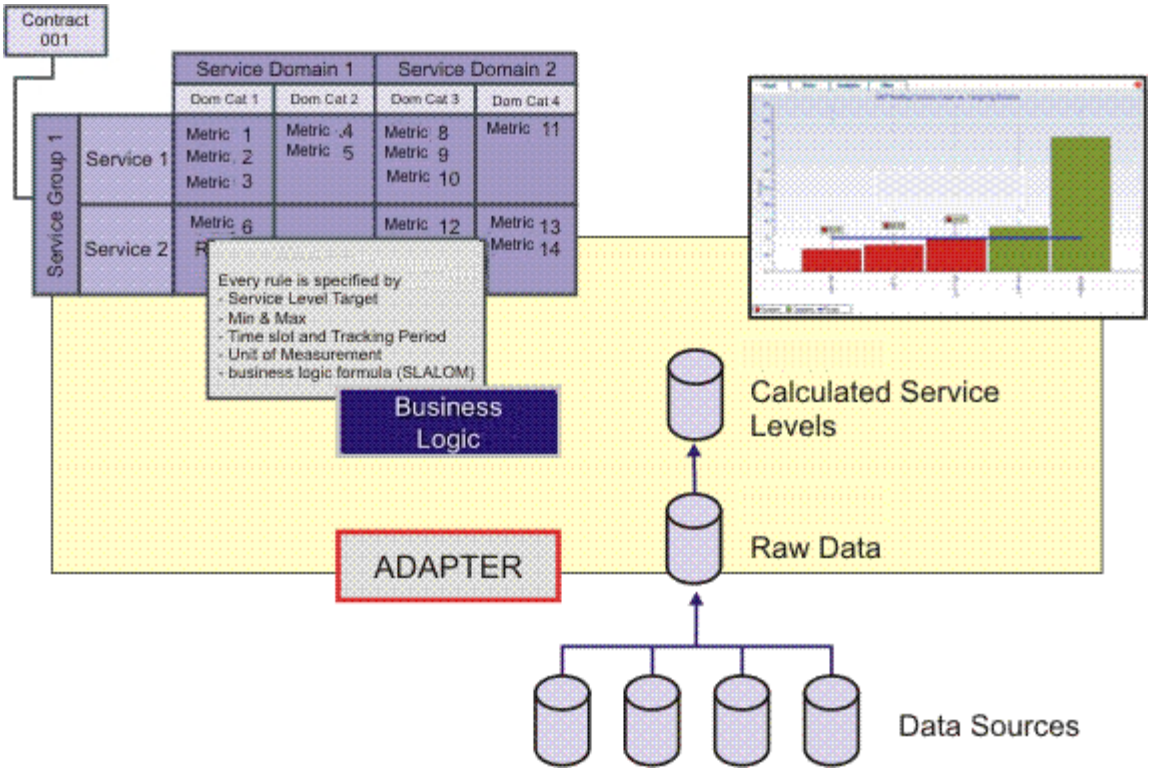
An available work-around is to add some commands to the beginning of the bat file that runs the Adapter. These commands copy the data sources that need to be loaded from the specific folder (according to their conventions) to a single dedicated folder that was created specifically for this purpose. The Adapter always loads the information from this folder.

The following image describes this solution:



Business Logic Scripting (Business Logic Expert)

The following image depicts the Business Logic position within CA Business Service Insight. It falls beneath each of the Metrics within the contracts.



At this stage of the implementation, the relevant Adapters are configured and the raw data records should already be available in the T_RAW_DATA table in CA Business Service Insight. It is now possible to apply the business logic to the events in order to produce the actual service level result for each Metric.

Business Logic Scripting is the process of writing code that logically operates on the raw data (Raw Data sent by Adapters) in order to calculate the service levels.

Each Metric has its own Business Logic formula with which to calculate the actual service level, although many of the Metrics in the system usually have a common logic that can be applied to different sets of raw data events.

For example, a Metric that calculates the Severity 1 ticket resolution time, and another Metric that calculates the Severity 2 ticket resolution time evaluate a different set of records-one uses only the Severity 1 tickets and the other only the Severity 2 tickets. However, both would most likely apply the same method in order to calculate the resolution time. The resolution time script will be developed and tested once, defined as a Business Logic module, and will then be used by both Metrics by including this module into the Metrics Business Logic areas.

Therefore usually when developing Business Logic scripts the main modules or templates are developed so as to make them available for all the Metrics in the system. In addition, each Domain category usually reflects a different type of measurement and therefore each Domain category will generally follow a different Business Logic module or template.

Business Logic Scripting Workflow

The business logic scripting stage involves performing the following steps:

- Define a formula

  Create the formula based on the calculation requirements defined in the design phase. The formulas defined are all the unique formulas to be used in their various permutations in the Contract's Metrics, each as a Business Logic module.

  For example, if the contract holds three Metrics for calculating the average ticket resolution time and one Metric for each ticket priority, then a single formula is developed for calculating the ticket resolution time, and having the ticket priority as its parameter. This formula, once tested, is defined as a module and is attached to all of the relevant Metrics.

- Test the formula

  Tests are performed to ensure that the formula is defined correctly and without errors, and the calculations produce the expected result. It is important to cover all extreme end cases and boundary conditions as part of the testing. The Business Logic scope is where the formula is executed for the purposes of testing. When initially defined, the formula is tested in its entirety. Then, once it is applied to all the Metrics as a module, it is important to run each Metric in the scope at least once to see that it receives Events (that is, that the registration is correct) and produces a reasonable result.

- Define the associated SLALOM module

  Each module is a unique Business Logic calculation and with its parameter definition can be applied to all the relevant Metrics. During the definition of the module, it is important that the module is thoroughly tested and documented in detail; what does the module do (calculation description), which parameters does it expect (name, meaning and use), etc.

- Attach all the Metrics to the relevant Business Logic module

  For each Metric in the defined Contracts, a link to the relevant Business Logic module should be defined. It should then be executed in the Business Logic scope to ensure that the link was implemented correctly and that the registration is working properly by receiving the relevant Events and producing the expected results.

Business Logic Modules

There are a number of important considerations when building modules for business logic, especially if you are using multiple modules within a single Metric:

- To ensure the use of a module is clear, you should add a comment at the top of the Business Logic for that Metric.
- If you are using a small piece of code within the Metric's business logic space, and including one of more modules in the code, you should ensure that for any default event handlers or subroutines, you remove that section of code from the main Metric business logic. You must make sure that each subroutine and event handler is only defined *once* in each of the modules being used by a particular Metric. This is to avoid confusing the calculation engine and producing unexpected results.

  Note: If, for example, you define the OnPeriodStart() function in your module and put specific code in it, and then leave the default OnPeriodStart() with no code in your Metric's main business logic screen, then at runtime the engine does not know which subroutine to execute. It may not execute the code that you are intending it to execute.
- You need to be extremely careful if you are parameterizing the OnRegistration subroutine. In some cases, doing so can break the automatic trigger built within the system to recalculate Metrics based on the addition of new raw data.

Inside the Business Logic

All Business Logic is located in an event-driven script, based on the syntax of VBScript. Each event that reaches the Business Logic, triggers an event handler.

The following types of events are sent by the engine to be evaluated by the Business Logic:

- Raw Data Events. Actual raw data inputs on which the Business Logic bases its results. The engine sends only the relevant Raw Data events based on the registration of the formula.
- Engine Events. Events sent by the engine implicitly that reflect the properties of the Metric definition, such as, timeslot definition and tracking period.
- Intermediate events. Events generated by other Business Logic scripts and can be used within another one.

The Business Logic formula contained within the Metric definition is what evaluates the events and outputs a service level result on which the reports are based. Depending on these service level results and the Domain definition, the engine also produces a deviation result (if a service level target has been applied to the Metric). The results produced are stored in a database table called T_PSL. It is this table that is queried by the Report Wizard when generating the reports, hence, all report data is precalculated to ensure that reporting performance is maximized.

Events Flow

As previously mentioned, the inputs to the Business Logic are the Engine Events and the Raw Data Events.

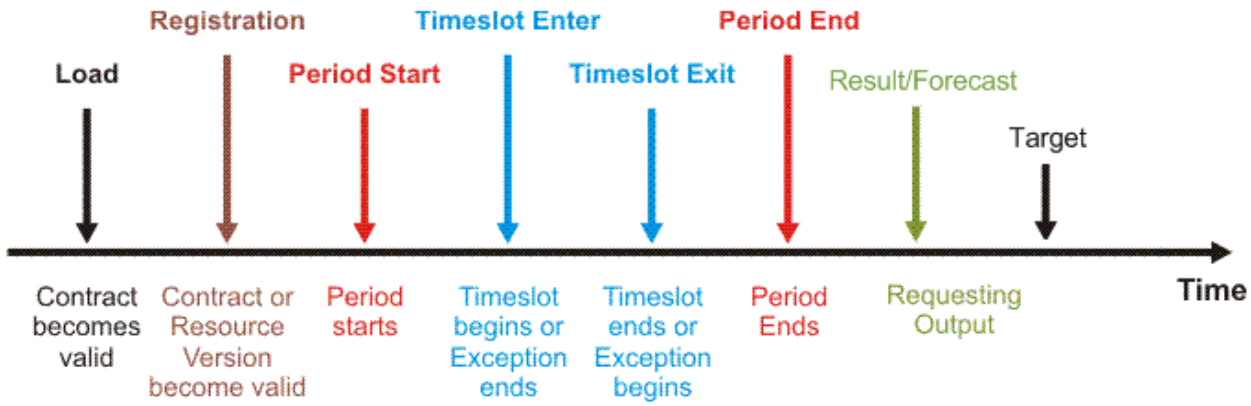
Raw Data Events received by the Business Logic are determined by the registration function within which the code requests a specific set of raw data events defined by their Event type and Resource identifiers.

In the Business Logic, registration also associates a user-defined subroutine that is executed in order to handle the Raw Data Event once an event is received. (By default, it is the OnXXXEvent, which should be renamed to something more meaningful.)

The Engine Events are triggered by the engine according to the associated Contract and Metric definitions. Once the Engine Event is triggered and received, the engine executes the pertinent event handler. Each Engine Event has an implicit event handler. These event handlers are functions and procedures defined on top of the VBScript. The event handler that handles the registration and the “result” function are both mandatory to implement in the code. All other event handlers are optional. However, the Business Logic does not deal with Engine Events for which event handlers are not implemented. Therefore, it is good practice to leave them all in place (even if not being used) to allow for future enhancements.

Note: When writing the Business Logic script it is important to implement all of the Engine events in order to be able to cover all the final possibilities. For example, even if during the first stage of the implementation a timeslot definition was not applicable, but will be in the future, then all the formulas will require amendment. It is therefore recommended for the Business Logic Expert to define the behavior "in timeslot" and "out of timeslot" periods in advance, even if initially it is not applicable, so that when the behavior is introduced, the required system changes are trivial.

Following are various Engine Events and their event handlers:



- **OnLoad (Time)**-(optional), called once at the start of calculations when the Contract is being activated. Can be used for initializing global variables.
- **OnRegistration (Dispatcher)**-(required), procedure for requesting the relevant Raw Data events and associating them with the user-defined procedures with which to be handled. Events are requested and are associated with procedures by using the methods of the Dispatcher object.
  OnRegistration is called once by the calculation engine at the beginning of the calculation of the Metric and again each time a Resource registered for the Metric becomes effective whereupon the Engine evaluates the set of changes done for that resource which can influence the set of Events streamed to the formula. The Engine has the definition of the Event request defined by the Event type and Resources identifiers and in a case where a Resource (or a change set containing multiple resources) changes something related to this set. Once it becomes effective, an OnRegistration event handler is triggered.
- **OnPeriodStart (Time)**-(optional), called at the beginning of the agent time period (set according to the time unit). The first OnPeriodStart is triggered once the Contract becomes effective, where the balance of the periods start with whole round hour time units. This event handler is usually used to periodically initialize variables, such as a counter that holds the number of incidents opened within the calculation period which should then be initialized to zero at the beginning of each period.
- **OnPeriodEnd (Time,IsCompleteRecord)**-(optional), called at the end of the agent time period (set according to the time unit). It is always called at the rounded end of the time unit in rounded hours (for example, 24:00). IsCompleteRecord is True when the period of the Metric has ended (according to the real world time of the application server), and is False when making an intermediate calculation. This event handler is usually used to do final calculations for the ending period in order to prepare the ground for the final result that will be supplied by the Result function.
- **OnTimeslotEnter (Time)**-(optional), called when entering a TimeSlot period based on the associated Metric definition. For example, if the Metric is associated with a timeslot definition of business hours where each day at 8:00 AM is a beginning of time slot, then at this time every day this event handler will be triggered.
- **OnTimeSlotExit (Time)**-(optional), called when exiting a TimeSlot period based on the associated Metric definition. For example, if the Metric is associated with a timeslot definition of business hours where each day ends at 17:00 PM, then at this time every day this event handler will be triggered.
- **Target()**-(optional), called when the Metric is specified as having a dynamic target. It allows you to determine the service level target of a Metric during business logic runtime. Such targets can include the consumption of service components or infrastructure changes. It has four values, just like the Result function; one for each mode. This function is executed *after* the Result function during normal execution.

- **Forecast()**-(optional), called once when performing a Commit of the contract version by having the calculation engine calculate the contract once in a forecast mode. The Forecast mode is performing a full calculation engine cycle on the contract (from the start to the end date of the contract version). The purpose of this cycle is to calculate the forecast values only. (There are no calculations are done on the T_RAW_DATA table).This function is called *in place* of the Result function during this execution mode.
- **Result()**-(required), called by the calculation engine in order to get the result of the calculation for a given period. The Result function is always called after the OnPeriodEnd event handler.

The following are the steps followed by the calculation mechanism (PslWriter service) in order to process a single Business Logic formula:

- PslWriter loads the formula into memory and executes every statement that exists in the declarations section (the declarations section is all the code which is outside of a function or sub-routine). Note also that all attached modules and libraries are included and compiled into this single code module for execution.
- PslWriter calls the OnLoad event handler.
- PslWriter calls OnRegistration.
- Calculation of the period begins with calling to OnPeriodStart.
- For each Raw Data Event registered during OnRegistration that falls within the time range of the period, PslWriter calls the user-defined event handler associated with that Event.
- If the beginning or end of the timeslot falls within the time range of the period, OnTimeslotEnter or OnTimeSlotExit will be called.
- If there is a relevant infrastructure change within the time range of the period, OnRegistration will be called.
- Calculation of the period ends with calling OnPeriodEnd.
- If a Dynamic Target has been specified, then this function will be called.
- The PslWriter calls the Result function in order to get the final result of the period calculation.

  Note: When the contract version is first committed and a forecast has been selected the Forecast function is called instead of the result function. This, however, occurs only this one time for each version)

During each calculation cycle, the Engine evaluates what the relevant Engine Events and Raw Data Events are based upon the calculation time period. It will first sort them by time and then send them to the relevant formulas for calculation. The time of the Raw Data Event is its timestamp and the time of the Engine Event is its triggered time. Both these types of events are then combined in a time sequence, and sent for calculation.

The timings of the Events are according to the associated local Metric, but the Time parameter of the event handlers (ie, OnPeriodStart (Time)) and the timestamp of the Raw Data Event is according to their value in UTC. The Engine compares them according to their values at UTC, so as to use a constant point of reference.

***Example*:**

If a Metric's timezone difference from UTC is two hours (i.e. GMT+2), and an incident opening event has its timestamp as 10:00 AM, then inside the Engine the timestamp that the event handler will be using is actually shifted accordingly and really starts at 8:00am UTC. Assuming the Adapter is configured to use this same timezone, then the raw data events will be shifted back 2 hours to UTC in the database also. When the events are being passed to the Business Logic, the calculation Agent responsible for calculation the Events for the period starting at 10:00 AM will actually be using the UTC time for events, which will be 8:00am onwards. However, if you use the out.log message in the code to print out the timestamps, it will show the timestamp shifted to UTC and hence will show 8:00 AM, despite the specified period (according to the Metric) being for 10am.

In the following calculation requirement it is important to convert the timestamp of the event before using it:

If a Metric is to calculate the number of incidents that were closed on the same day that they were opened, then it is necessary to compare each incident's opening and closing time. If an incident's opening and closing time fall on the same day (and within a defined timeslot), the incident will be counted.

It may happen that during the process of shifting the incident to UTC from its original local time the day will change.(ie. again using GMT+2). An incident opened at 1am, will be shifted back to 11pm the previous day in UTC) It will then not be counted even when it should have been. This is a situation where the time has to first be converted back to the local Metric time, and only then be compared. For such a case, use the GetLocaleTime(utcTime) method. This method converts a time given in an UTC time zone into the time zone of the local Metric.

```
The following is the event handler code:
Sub OnIncidentEvent(eventDetails)
   If dateDiff("d",Tools.GetLocaleTime(eventDetails.time),_
     Tools.GetLocaleTime(eventDetails("ClosedAt")))=0 then
          CountIncidents=CountIncidents+1
   End If
End Sub
```

Registration

Registration is the process whereby the Business Logic submits a request to the calculation engine for the set of Raw Data Events that are to become a part of the calculation.

The registration process can be handled in two ways; Using the Registration Wizard or manually using the dispatcher object within the Business Logic.

Using the Registration Wizard is a simple process of choosing from the available options. You have all of the same options available as when doing the registration manually, without the ability to use parameters. If you need to use parameters, you will have to perform the registration manually. The basic flow of the wizard requires you to first determine which type of registration you would like to perform, then set the resource types and events upon which registration should be performed, and finally which event handler will be used to process the collected events.

Once the registrations are completed, they will appear listed in the 'Registration' tab of the Metric. Note also that it is possible to have more than one registration statement for a Metric.

In effect the Registration wizard uses the same functionality as the Manual registration, and all these options are discussed in the following section.

When performed manually within the Business Logic, the registration of the formula is handled by the *OnRegistration* event handler. This must be implemented in the formula and triggered whenever a Registration Engine Event is triggered. The Registration Event is triggered once when the Contract is activated, then every time a relevant resource or Change Set becomes active. A change in the affected resource is said to be relevant if it affects the events which the Metric is meant to receive. For example, if registration is done by Contract Party (*RegisterByContractParty*), this means that all the events of the defined type whose resources are attached to the Metric's Contract Party are a part of the calculation. In such a case, every time a new resource is attached or detached to that contract party, the registration method will be triggered to notify the Engine of the change.

The registration methods are provided by the Dispatcher object which is passed to OnRegistration as an argument. The different methods provide various ways in which to define the filtering criteria based on the event type definition and any resource allocation criteria, such as, resources of a resource group, or Resources of a certain type.

Using the Contract Party and Service registration methods is highly recommended because it makes it easier to use the Business Logic as a module or template. When doing so, the relevant Contract Party and Service is taken from the associated Metric definition and when re-using the formula for different contracts and/or service components, the registration does not need to change.

Another popular registration method is RegisterByResourceGroup, which is handy for working with resources that are logically grouped but may not always be associated with Contract Parties or Services. The assignment of resources to the groups here can then be managed by the resource catalog (individually or via Change Sets) and could even be updated automatically by translation scripts.

In general, the registration method is determined during the design phase and is directly driven by the defined data model.

Note: In order to check whether the Dispatcher object has been correctly used, the OnRegistration function is also called during the syntax checking of the SLALOM. For this reason, one should not assume that the OnLoad was run before the OnRegistration function and one should not use some of the properties of the context object, such as, "TimeUnit", "IntervalLength", "IsPeriod", "CorrectionsApply", and "ExceptionsApply" in the OnRegistration event-handler.

The registration methods are also responsible for associating the events with a procedure that will be triggered according to the event's timestamp. The procedure defined can have any name, but will always have the eventDetails object as its parameter. The eventDetails object reflects the Raw Data Event received and holds all of the event details as data fields, as shown in the following registration:

```
Sub OnRegistration(dispatcher)
dispatcher.RegisterByContractPartyAndService "OnMemUseEvent", "MemUse", "Server"
'the parameters of the method are: <procedure name>, <Event Type>, <Resource Type>
End Sub
```

The above registration statement tells us that all Raw Data Events of the 'MemUse' event type, and associated with the 'Server' Resource Type, will be sent to the 'OnMemUseEvent' event handler in the Business Logic.

The following procedure will also have to be defined ahead in the Business Logic:

```
Sub OnMemUseEvent(eventDetails)
 If InTimeSlot And eventDetails("MemoryUsage")>MaxMemoryUse Then
       MaxMsmoryUse = eventDetails("MemoryUsage)"
     End If
End Sub
```

By referring to the eventDetails object and using the 'MemoryUsage' parameter, the statement extracts the value of the MemoryUsage field from the event which was passed into the function. These fields are the same ones defined in the event type, and the field names are case sensitive.

Register Clustered Metrics

Clustered Metrics enable the definition of one Metric to run for each individual member of a resource group, to apply the same definition and logic for a set of items. A clustering can be set either statically on a predefined set of resources or dynamically on the members of resource group while the group can be changed over time and include or exclude members from the group.

Note: For a detailed description see Appendix E - Defining Business Logic Formulas (Business Logic Expert).

Clustered Metrics are used when it is necessary to calculate a service level result for each item in a group of resources. The items in a resource group can be either resources or other resource groups, therefore the registration method in a Business Logic script of a clustered Metric must be RegisterByResourceGroup or RegisterByResource, where the resource name or resource group name specified is defined as the item in the cluster. This is done by using the 'ClusterItem' property of the context object which holds the name of the current cluster item.

***Example*:**

```
dispatcher.RegisterByResource  “<ProcedureName>”, “<Event Type name>”,
Context.ClusterItem
```

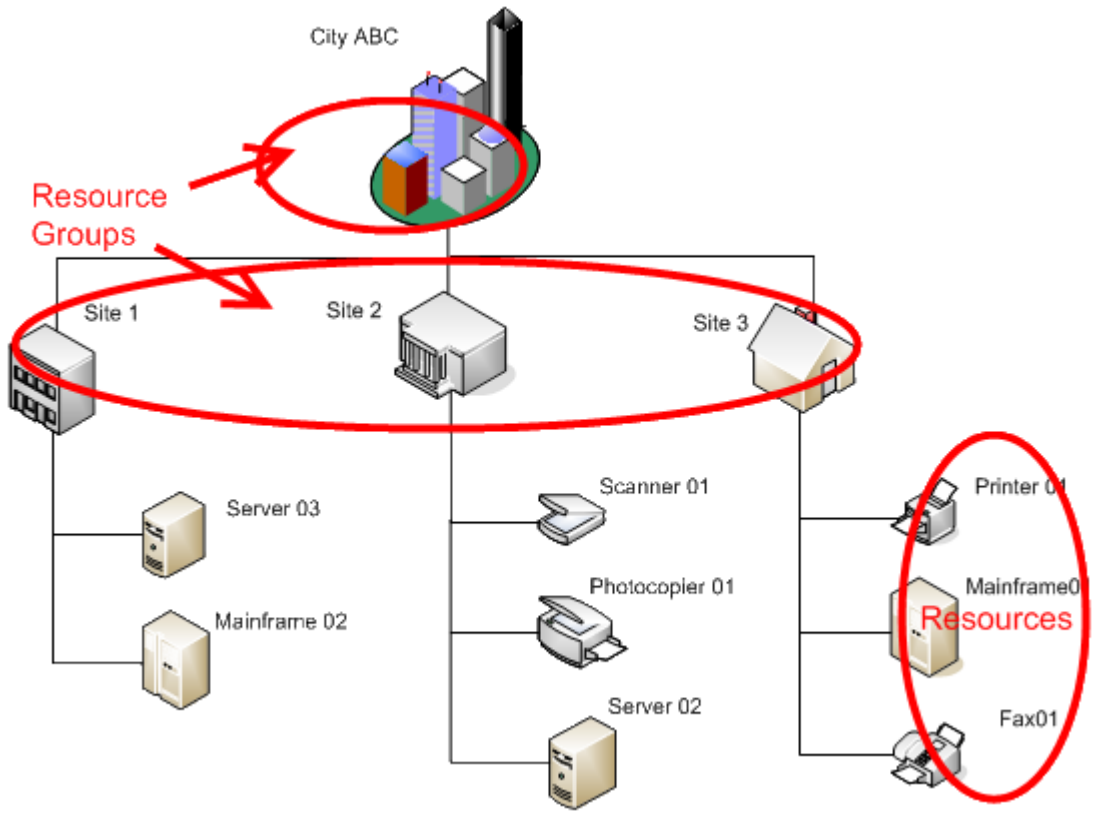
In cases where this registration method is used, the Metric calculates a result for each of the resources in the resource group over which the Metric is clustered,

-OR-

```
dispatcher.RegisterByResourceGroup "<ProcedureName>", "<Event Type name>",
Context.ClusterItem
```

In cases where this registration method is used the Metric will calculate a result for each of the Resource Groups contained within the Resource group for which the Metric is clustered over.

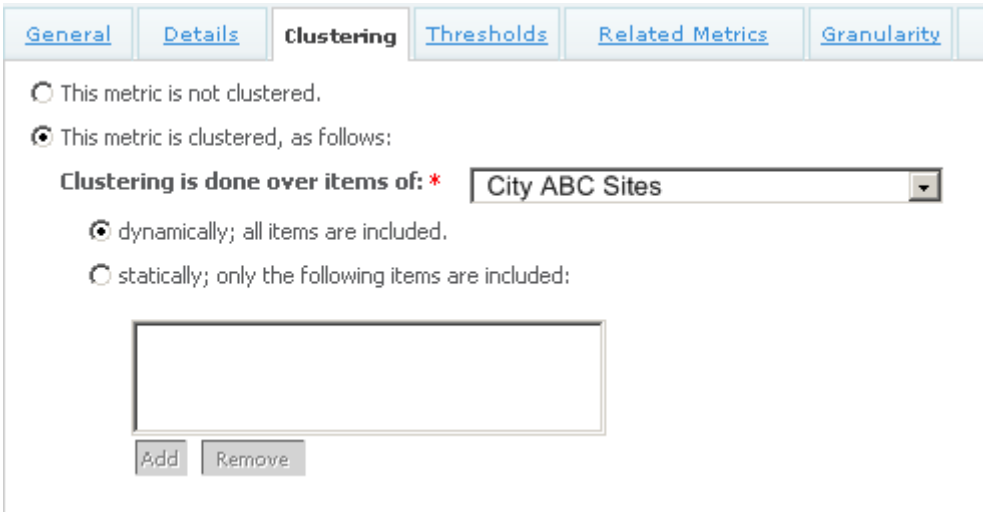
Clustering can take place at different levels depending on how the resource model is created. It is often the case that organizations have different layers of grouping which they wish to represent. For example, within a particular city, there may be a number of sites, and within each site there could be a number of infrastructure devices. (Printers, Scanners, Servers etc) Using this type of grouping you could structure a defined resource hierarchy which contains multiple levels and groupings of these hardware items. (Assuming that an infrastructure device will be the 'resource'. The structure described here could look as follows:


City ABC
Resource Groups
Site 1
Site 2
Site 3
Server 03
Mainframe 02
Scanner 01
Photocopier 01
Server 02
Printer 01
Mainframe0
Resources
Fax01

As can be seen from the diagram, there are multiple layers of groups. The top level 'City ABC' group contains three different sites (which are also resource groups). The resource group 'Site 3 Resources' contains three different resources. So from the previous example, to cluster the Metric across the three different sites, you would use the following registration:

```
dispatcher.RegisterByResourceGroup  "<ProcedureName>", "<Event Type name>",
Context.ClusterItem
```

In this case the Context.ClusterItem refers to the resource group called 'City ABC Sites', which contains three other resource groups called 'Site 01 Resources', 'Site 2 Resources' etc, and may look like this on the Clustering tab of the Metric.



Notice also that the clustering is set to dynamic since this will automatically include any changes to the Group as they occur. Static clustering can be useful for subsets of resource groups or when you do not want the clustering to change over time.

To create a Metric that reports on the resources of the Site 3 group, use the following registration statement:

```
dispatcher.RegisterByResource  “<ProcedureName>”, “<Event Type name>”,
Context.ClusterItem
```

In this case, the Context.ClusterItem refers to the individual resources, hence register by resource only. The Clustering tab of the Metric contains a reference to the 'Sites 03 Resources' group.

You can configure clustering to operate on different levels of the hierarchy within a single Metric. For example, assuming the situation described in the previous example and clustering this Metric across the 'City ABC Sites' group again. You can include within one Metric, the resource members from different levels of the hierarchy. In this case there are three options as to which resources are included in this grouping:

- **First Level only : Direct members**: Same as the older cluster methods described previously.
- **All Levels: Include Resources Only**: Includes all resources contained in the three sites resource groups in a single level, and calculates the service level for them individually.
- **All Levels: Include Resources and Resource Group**: Includes all resources contained in the three sites resource groups, as well as the three resources groups, and calculates the service level for them individually.

Agents

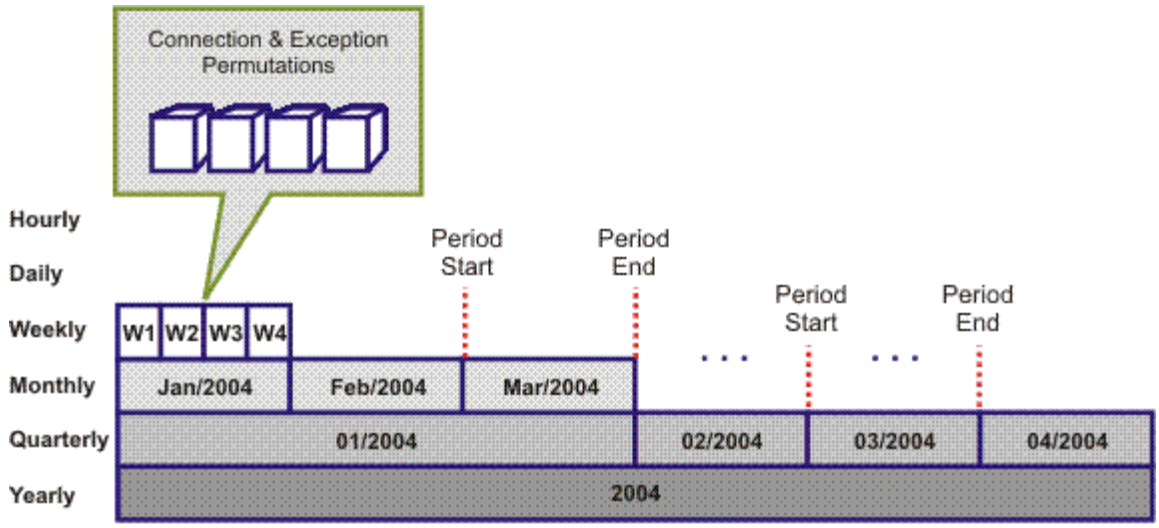
Each Metric has a definition of a tracking period. The tracking period is the period during which the Metric has to output a service level result and it is this result that should be compared against the defined target. The result produced for the tracking period is the business result, in other words, it is the contractual result for which the provider has committed to provide a certain target service level. Each instance of the Business Logic for each Time Period is known as an Agent, and is directly related to the granularities associated with each Metric.

For example, if the Metric is defined with a one month tracking period, then the Metric is executed to provide a monthly result.

In order to provide the drill down capability where the user can drill down within the monthly result to see the daily result, the Metric must have a definition of additional time units in which it should be calculated. The time units are defined in the General details section of the Metric in the Granularity tab.

For each of the time units defined in the Granularity tab of the Metric and for the tracking period, a separate Business Logic instance is executed by the Engine. Each of these instances executes the same Business Logic code, but the OnPeriodStart and OnPeriodEnd will be triggered differently for each of these instances. For the daily instance it will be triggered at a start of day and at the end of day. For each time unit it will be triggered according to the time unit start and end points.

Each of the Business Logic instances is executed in order to produce the relevant time unit result. In addition, each period has to have a result that takes any exceptions into account. Any period that does not do so, (if exceptions are defined) must allow the user to choose whether to see the service level result with or without the exceptions that were taken into consideration. Due to this requirement, each Metric will potentially have fourteen Agents (instances) executed by the Engine, two Agents for the business results and twelve for the six additional operational periods.



Business Logic Time scale

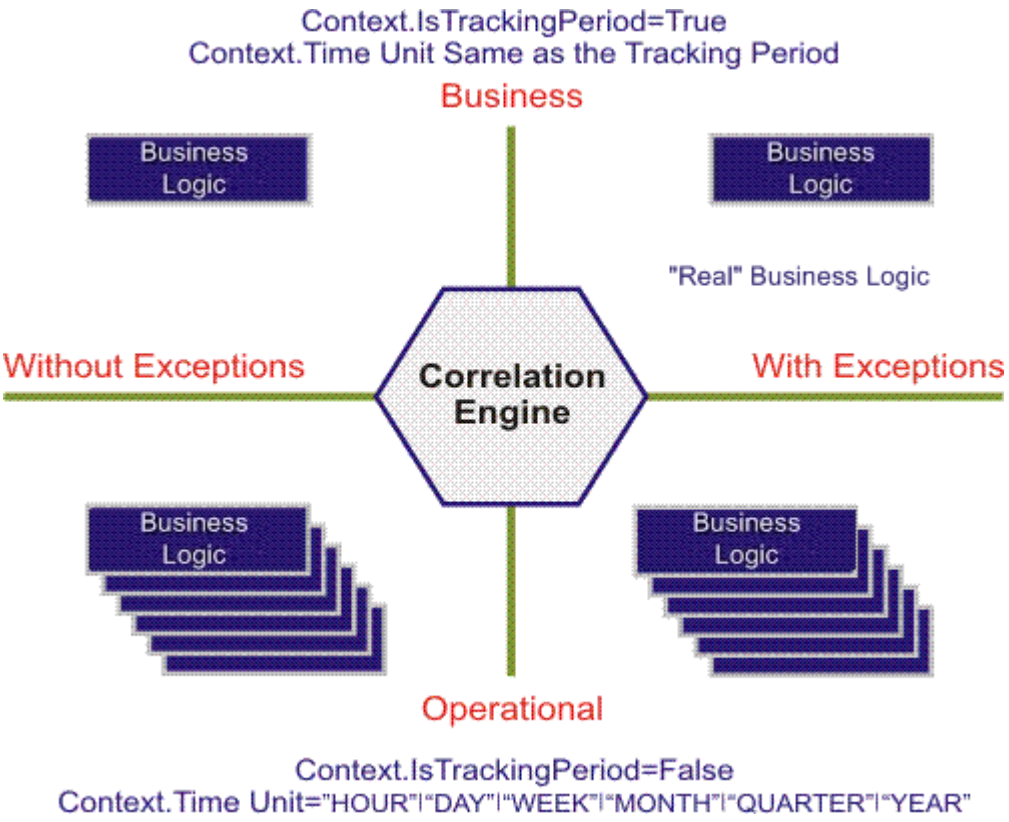
This means that the granularity definition has a serious impact on the calculation Engine performance because each time period is calculated for a different Agent. So, in cases where the drill down capability is not required as a whole or partly, it is recommended that some of the Agents are turned off. This has an especially large impact in the case of the lower granularities such as hourly results. It also has an enormous impact for Clustered Metric, since the Engine does all of the above calculations for each ClusterItem it encounters. In effect it treats each ClusterItem as a new Metric. For example, when calculating a Metric across a resource group of 50 items, there will be 49 times more work for the Engine to do, compared with the same non-clustered Metric.

For example, if the Metric is set to calculate the resolution time in number of days, then an hourly result is meaningless, and should be unchecked on the granularities tab to avoid the Engine performing needless calculations.

The TimeUnit attribute of the Context object (i.e context.TimeUnit in the Business Logic) returns the time unit of the agent currently executing, where the possible returned values are: HOUR, DAY, WEEK, MONTH, QUARTER, YEAR.

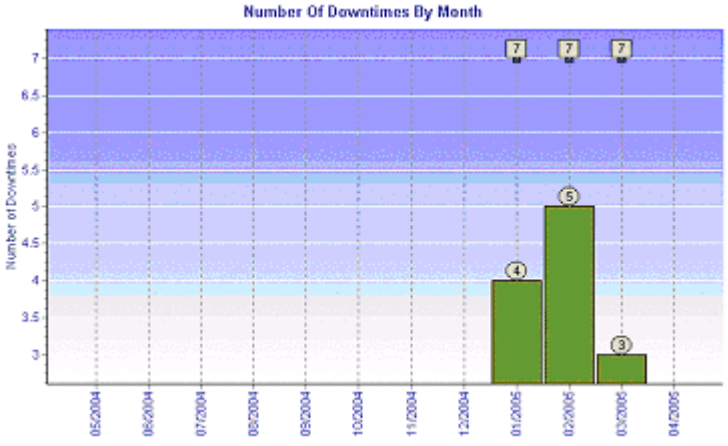
For example, for the daily Agent the Context.TimeUnit will return "DAY".

The IsTrackingPeriod attribute of the Context object will return True for the Agent that calculates the tracking period time unit. Its also important to note that the Agents shown in the Granularity tab of the Metrics are additional to the Agent of the tracking period of the Metric. So, even for a Metric with a monthly tracking period you can turn off the monthly agent and it will still calculate the monthly service level, but only as 'Business results'.(Non-operational results)

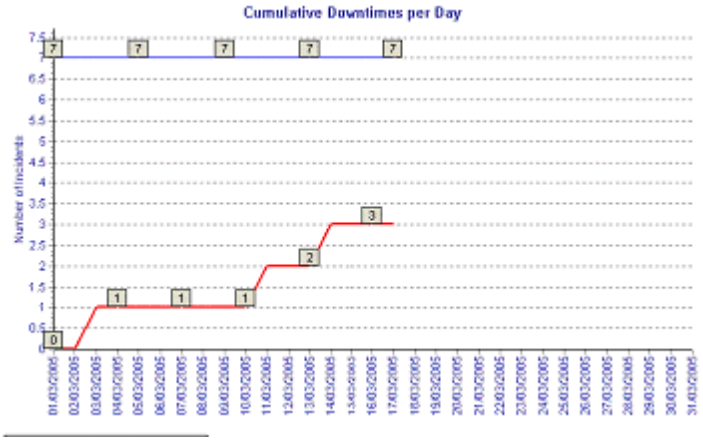


As mentioned above, the various Agents usually execute the same Business Logic code, but there are cases in which it is necessary to have a slightly different logic applied.

For example, in the monthly case, the result should be the number of down times for each month separately as shown below:



For the daily drill-down it may be necessary to be able see the cumulative down times, where each day is a sum of all the days from the beginning of the month. This method should be applied to all time units smaller then a single month as shown below:



The difference between the two time units is that for the Agent calculating the tracking period the down time counter will be initialized to 0 at the beginning of every period, but for the daily Agent the counter will be initialized only in a case where the day is the first day of the month.

```
The following is the OnPeriodStart event handler:
Sub OnPeriodStart(time)
     If InStr("MONTH,QUARTER,YEAR", Context.TimeUnit) > 0 _
 Or (Context.TimeUnit = "DAY" And Day(time) = 1) _
 Or (Context.TimeUnit = "HOUR" And Day(time) = 1 And Hour(time) = 0) Then
          DownTimeCounter = 0
     End If
End Sub
```

What is recalculation?

Recalculation is performed when the Correlation engine identifies that a Metric's final periodic result is no longer valid and it must therefore recalculate the results.

Recalculation may be caused by:

- Receiving new events that actually occurred in the past. (earlier than when the Engine has calculated *up to presently* for that Metric)
- A Resource which is registered to the Metric is altered (new version date and changes made to it).
- Committing a new contract version which overlaps the previously calculated time with changes to some Metrics (only changed Metrics are recalculated)

The most efficient method for registration is by Contract Party and Service. Arranging the resources under Contract Party and Services is a way in which to express the logical relationship between the data layer and the business layer in the system. Registering resources through these entities will require no changes to the formulas when used in different Contracts or when they are used for different Services. The Metric's current context identified the contract and service, which defines the relevant Contract Party and associated Service. The Business Logic formulas that are defined in this type of registration are easily reusable because they do not need any changes in the registration.

Outputs - User Tables

The standard Business Logic script does not have access to external output tables. There are only two output destinations:

- The PSL table (T_PSL), to which the Engine automatically writes the service level results, and where the result specified in the result function is written to. Service level values that are written to this table can be only of a numeric type. The results written to the T_PSL table are the results that the report wizard reports on. There is no control over the structure or method that these results are written in. This is done automatically by the Correlation Engine.
- The SLALOM Output table (T_SLALOM_OUTPUTS). Writing to this table is done using the methods provided by the Tools Object from within the Business Logic formula. When outputting to this table, a logical table name is provided which is referred to as the User Table. These tables are used to output information during the service-level calculation. This information can later be used to generate freeform reports. There may be many user tables.

The use of the T_SLALOM_OUTPUTS external table is required when additional output is needed over and above the periodic service level result, when the additional output cannot be provided by adding another Metric, or when adding another Metric decreases the calculation performance by going through the same set of records, only to provide a different output.

For example, consider a case where a Metric is set to calculate the percentage of tickets that were resolved in less than a day and a report with the list of all the tickets that failed to be resolved in less than a day should be produced, it is necessary that the formula output each ticket that is found as a failure to an external table, in addition to adding it to the calculation statistics.

With the above requirement, the normal output service level table cannot provide this output, because:

- The service results are all numeric
- Only a single service level result is possible for each time period.

Records are written to the user output tables only for the Agent which is running for the tracking period of the Metric and calculating the exceptions and corrections. For example, if the Metric is defined as having a monthly tracking period, then the output statements (Tools.SaveRecord and Tools.SaveFields) do NOT get executed when the Engine is executing the formula for the other granularities such as HOUR, DAY, WEEK, QUARTER and YEAR

An additional benefit of having this table externally is that it can be used for multiple reporting requirements. Other typical reporting requirements can be generated from these tables in addition to the contractual requirements. For example, a Metric calculating the 'Number of Tickets Closed' in a month, could also be calculating the ticket resolution time and outputting all this information to the output table. This will enable a more detailed analysis of the individual tickets that were closed during the period, along with additional details that may be required under a separate reporting requirement.

The information in these tables is also managed by the Engine recalculation mechanism. During this recalculation process, records are marked as inactive, and a new row is written instead. This is an important point to note since all freeform report SQL statements need to include a check on the IS_ACTIVE field in the T_SLALOM_OUTPUTS table (where is_active = 1) since only these records are current.

Note: When running the Business Logic Scope during the debugging process of the formulas, messages are actually written to the table T_DEBUG_SLALOM_OUTPUTS instead of the T_SLALOM_OUTPUTS table.

When documenting data using T_SLALOM_OUTPUTS, the inserted data is always text (since the fields of T_SLALOM_OUTPUTS are all varchar2). Therefore, date values are converted into text by applying the operating system's format which may change during the application lifecycle. T_SLALOM_OUTPUTS may therefore suffer from inconsistencies in date formats. Moreover, the Business Logic handles UTC dates, where one may expect T_SLALOM_OUTPUTS to hold local timestamps, so in some cases it may be necessary to use the conversion function *Tools.GetLocaleDate(date)* to work around this.

```
The following function converts dates into local times and keeps the date format
consistency by formatting dates into the "dd/mm/yyyy hh24:mi:ss" format:
Function FormatDate(time)
     Dim LocalTime
     LocalTime=Tools.GetLocaleTime(time)
     FormatDate=Day(LocalTime) & "/" & Month(LocalTime) & "/" &
  Year(LocalTime) & " " & _
     Hour(LocalTime) & ":" & Minute(LocalTime) & ":" & Second(LocalTime)
End Function
```

There are two methods that can be used to write to the external table from within the Business Logic formula which are:

- Tools.SaveRecord <parameter list>
- Tools.SaveFields <parameter list>

Both of these methods of the Tools object are described below in more details:

```
Tools.SaveRecord tableName, key,[val1],[val2],…
```

This method saves a record in a table called T_SLALOM_OUTPUTS. The tableName parameter specifies the (virtual) table within T_SLALOM_OUTPUTS to which the information should be written. Each record in the user table has a unique key which specifies the record to which the information should be written. Each record also has up to 20 string-type value fields. The SaveRecord method receives a user table name and a key. It also accepts all the value fields in the user table. (These value parameters are optional and may be omitted.) If a record with the same key already exists, it is updated. (Only the value fields transferred as parameters are updated.) If a record with this key does not exist, it is created.

```
Tools.SaveFields tableName, key, [fieldName1,fieldVal1], [fieldName2,fieldVal2]
```

This method is similar to SaveRecord, except that instead of enumerating all values, it provides pairs of field names and related field values. Field numbers can be substituted for field names. The fields names should previously be manually defined in the T_SO_FIELD_NAMES table. This table is used to record the structure of the output tables.

It is recommended that the Business Logic Expert define the structure of the output table before writing to T_SLALOM_OUTPUTS because then the structure and meaning of the fields is already well defined and will make the querying far simpler.

The table structure is:

- **table_name**-Each logical table has a unique name
- **field_name**-Each field in a table is unique
- **field_id**-Each field is numbered with a serial number starting from 1

Using the SaveFields method is preferable as it keeps a documentation of the structure and the meaning of the inserted values.

***Example*:**

Consider a case where it is necessary to produce a list of all of the incidents with the resolution time higher than a defined threshold, in addition to the Metric result calculating the percentage of tickets that were resolved below that threshold. Writing to the output tables will be done in the OnXXXEvent event handler procedure after the threshold validation.

This is illustrated by the following example:

```
Sub OnIncidentEvent(eventDetails)

    If eventDetails("RESOLUTION_TIME")<=SThreshold Then
     CountSuccessIncident s= CountSuccessIncidents+1
    Else
building the record unique key
     KeyString= eventDetails("IncidentID") &eventDetails.Time

        Tools.SaveFields "IncidentsTable", KeyString, "CASE_ID",
```

```
    eventDetails("CASE_ID"),_
           "CUSTOMER_REF",eventDetails("CUSTOMER_REF"),_
           "TICKET_CREATOR",eventDetails("TICKET_CREATOR"),_
           "CREATION_TIME",eventDetails("CREATION_TIME"),_
           "SEVERITY",eventDetails("SEVERITY"),_
           " RESOLUTION_TIME ",eventDetails("RESOLUTION_TIME "),_
           "CLOSE_TIME",eventDetails("CLOSE_TIME")
    End If
End Sub
```

A few suggestions related to the use of the T_SLALOM_OUTPUTS tables are as follows:

- It is recommended to write only the necessary data to the Output Tables (not more).
- Write to the Output Tables for only Metric granularities which are necessary.
- All of the value fields are only 50 characters in size by default (except for the column VAL_1 Which is 512) You may need to truncate certain values for the fields to ensure they fit this data size since otherwise you were get runtime errors in the Business Logic.
- There are only 20 columns available for your data (VAL_1 … VAL_20)

Note: Writing to the output tables may have an influence on the performance of the Engine, since writing to a table is computationally intensive, compared to a calculation in memory. Therefore consider carefully whether it is necessary to use this functionality and where it is necessary, and then minimize the number of times the tables are accessed.

Reporting on Information from the User Defined Tables

It is not possible to use the standard CA Business Service Insight Report Wizard in order to report on the information written to the output tables. In order to report on this information it is necessary to build a free-form report. In essence, this means building a SQL query on top of this table. Because this table contains many logical tables that are being written to by various formulas, it is recommended that someone familiar with SQL (Data Source Expert) creates a view over the T_SLALOM_OUTPUTS in order to make it easy to differentiate between the different logical tables contained within it, and also to make sure that the information is retrieved as planned.

The view definition will already have all of the casting of the string field types to the relevant information type and will also hold all of the filtering required (Logical table, active records, relevant Metric, etc.).

The following is an example of the view creation:

```
create or replace view kpi_view as
select
     to_date(t...) as fieldName,
     to_number(t..), …
from t_slalom_outputs t,
```

```
      t_rules r,
      t_sla_versions sv,
      t_slas s,
where table_name = 'TableName'
      and is_active = 1
      and t.rule_id = r.psl_rule_id
      and r.sla_version_id = sv.sla_version_id
      and sv.sla_id = s.sla_id
      and sv.status = 'EFFECTIVE'
```

Create Business Logic Modules

Users can define independent business logic modules that can be used by multiple service level objectives (Metrics). To apply system-wide changes to business logic, the user changes the "base" logic component which can then be applied to all relevant SLAs with a single click.

A Business Logic Module is a component of code within which business logic can be defined and easily maintained, and which reduces redundancy. A single module can be used by multiple Metrics.

During the configuration stage, the formulas are configured to define the main Business Logic modules. (See the Design chapter: Business Logic templates and modules.)

There are three types of Business Logic Modules:

- **Full Business Logic**-used when a whole formula is required to be used as a module. The Result and OnRegistration methods must be implemented in the Module script.
- **Class**-module that contains a definition of a single VB class.
- **Library**-module that contains any collection of procedures, functions and classes.

Modules can be used from within any of the following:

- Metrics
- Other Modules
- Business Logic Templates
- Metrics inside Contract templates and service level templates.

Modules can use parameters which are driven from the Metric context Parameters("ParamName").

Note: To avoid runtime errors, always set a default value when using parameters in Business Logic Modules. The formula outputs a log error message for non-existing parameters.

```
If Parameters.IsExist("ReportedDowntimesNum") Then
    maxBufferSize = Parameters("ReportedDowntimesNum")
Else
    maxBufferSize = 3
    out.log "ReportedDowntimesNum parameter not set", "E"
End If
```

Business Logic Module Example

There is a service level object described as "helpdesk activity within threshold". The following helpdesk system has a ticket lifecycle with statuses of:

- Open

- Assigned
- Closed.

Two of the Metrics that can be defined in order to describe the helpdesk performance are:

- **Metric name**- **Successful Ticket resolution on time**.

  Objective statement-no less than 99% of the tickets should be resolved within 4 hours.

  Business logic-Resolution should be calculated from Open to Closed.

- **Metric name**-**Successful ticket assignment on time**.

  Objective statement-no less than 99% of the tickets should be assigned within 30 minutes.

  Business logic-assignment should be calculated from Open to Assigned.

Since the same logic can be identified for both of these Metrics, a module can be created to cater for both Metrics.

The module requires the following parameters within the Metric context:

- First status
- Second status
- Threshold.

Once the Business Logic module has been created, the defined Metrics can use a module as part of the definition.

Next, the logic can be changed. For example, a new status of "Customer Pending" should be considered. Customer Pending is set for a ticket when the helpdesk is waiting for additional information about the ticket from the customer. Within the business logic, for the duration of time that the ticket is in 'customer pending' it should not be considered as part of the calculation. Only the Business Logic module therefore needs to be changed to take into account the new status and logic. A new version of the module including the new logic is created.

When committing the change, you are prompted with a list of Metrics that employ the module. You can either apply the change to all Metrics collectively, or choose to apply the change to only specific Metrics in the list.

If you select only specific Metrics from the list, you are prompted to create a new module for the selected Metrics. The old module used by the selected Metrics is replaced by the new Business Logic module and the recalculation is done using the new logic.

Create Parameters

Parameters provide business users a quick and intuitive way with which to view and change their values using the graphical user interface (GUI) without having to edit the formula script.

The use of parameters within the Business Logic enables the creation of general formulas that can have a wide use across the system and optimizes the use of the modules.

Parameters can be defined at Contract or Metric level. Metric level parameters are displayed and configured in the Objective statement tab of the Metric details. The Business Logic has access only to the Metric level parameters; therefore in order to access a Contract parameter from within a Metric, a different type of Parameter is created locally within the Metric. It is called a dynamic Parameter and it takes its value as a reference from the Contract level parameters. The reference values allowed in the dynamic parameter are only those defined in the parent contract of the Metric.

Parameters types:

- Date-in Date Modal (Date + Time).
- Number-maximum size of 15 digits with a maximum precision of 5 digits.
- Text-maximum size of 100 characters.
- Table-maximum size of 120 entries.

In order to access the parameter's values from the formula code, it is necessary to use the Parameters object and refer to the parameter name.

***Example*:**

```
Parameters("Threshold")
(Note this is a short-hand method of calling the value - normally this is done as
Parameters.Item("Threshold") )
```

Or, for a table type parameter:

Parameters("Table")("Word")("Price")

(where the "Word" and "Price" values are the row and column names of the tabled parameter, respectively)

Table parameters should only be used following a number of key points:

1. Define a global variable (e.g. dim myTableParam)
2. In the function OnLoad, fill the variable from the parameters object (e.g. “Set myTableParam = Parameters(“MyParametersTable”))

3. Thereafter use only the created object, myTableParam. The parameter itself should not be used outside the OnLoad function, and you should only refer to the global variable object you created from it.
   (e.g. “dim myVal: myVal = myTableParam (“myRow”)(“myColumn”)).

This frees memory that the parameter takes up and prevents the Engine from creating the parameter's map on each parameter call and for each "OnXXXEvent", which can be called thousands of times per Metric

The following code illustrates the correct usage of a table parameter:

```
Option Explicit

Dim sum
Dim myParamTable

Sub OnLoad(TIME)
   Set myParamTable = Parameters("MyTableParam")
End Sub

Sub OnRegistration(dispatcher)
  dispatcher.RegisterByResource" OnEvent", "EventType", "ResourceType"
End Sub

Sub OnPeriodStart(TIME)
   sum = 0
End Sub

Sub OnEvent(eventDetails)
   If Context.IsWithinTimeslot Then
      sum = sum + 1 * myParTimeSlotamTable("firstRow")("secondColumn")
   End If
End Sub

Function Result
   Result = ( sum * myParamTable("secondRow")("thirdColumn")  )
End Function
```

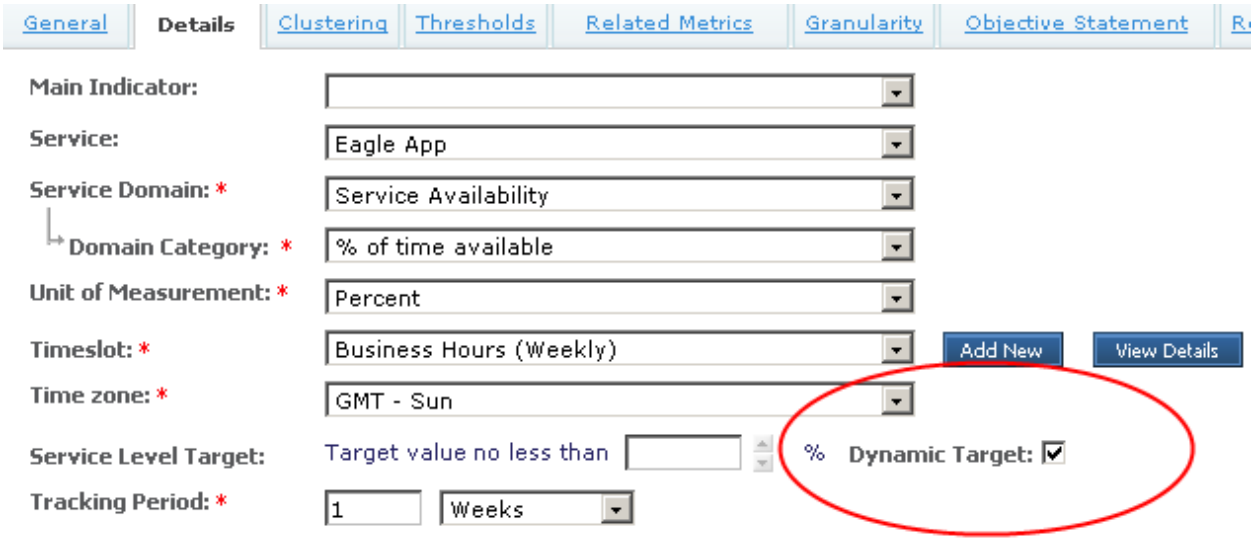
The following methods are available for each parameter object created within the code.

Parameter	Description
IsExist	Does param exist. Does not work on contract parameters.
IsNumeric	Is param of type Number.
IsText	Is param of type text.
IsDate	Is param of type Date.
IsTable	Is param of type Table.

IsEntryExist	Does an entry in Table exist.
IsEntryNumeric	Is entry in Table of type Numeric.
IsEntryText	Is entry in Table of type Text.
IsEntryDate	Is entry in table of type Date.
Dump	Returns a list of all parameters.
Item	Refers to the parameter value.

Implement Dynamic Targets

Dynamic targets are handled by the Business Logic using an event handler in the standard Business Logic script, similar to the Result() function which is used to the return the service level value from the Metric. The Dynamic target must be specified on the Metrics 'details' tab as shown below.



When a dynamic target is specified then the target is taken from the 'Target()' function in the Business Logic, rather than the static value specified in the Details tab of the Metric. The Target function looks like the following.

```
Function Target
  'TODO: ADD code here TO handle dynamic target calculation
  Target = Null
End Function
```

This function should be implemented based upon the requirement for the metric so as to return the desired target value for a specific period. The function can return any number that the business logic can assign to it.

A Real World Example of Dynamic Targets

For a Call Centre, the target for a Metric measuring the '*Avg Call Pickup time*' may depend on the volume of calls. If there are between 0 and 800 calls the target should be < 15 seconds, if there are between 801 and 1500 calls, the target should be < 20 seconds, higher than 1500 calls the target should be less than 25 seconds. This could be implemented as follows: (assuming TotalCalls is a counter incremented for each call event received and that TotalCalls cannot be less than 0)

```
Function Target
  If TotalCalls >0 and TotalCalls <= 800 Then
     Target = 15
  ElseIf Total Calls > 800 and TotalCalls <= 1500 Then
     Target = 20
  Else
      Target = 25
  End If
End Function
```

Another Example of how a Dynamic Target can be Used

Consider the situation where the target for a Metric may change depending on the granularity of the calculation. It may be the case that there is a daily target of 98% availability for a group of servers, but the monthly target is 99.5% availability. The solution for this requires using the dynamic target function in conjunction with the function call to *Context.TimeUnit* to determine the current agent which is being calculated. Hence you can adjust the target accordingly.

```
Function Target
  If Context.TimeUnit = “DAY” Then
     Target = 98
  ElseIf
     Target = 99.5
  End If
End Function
```

Back up States

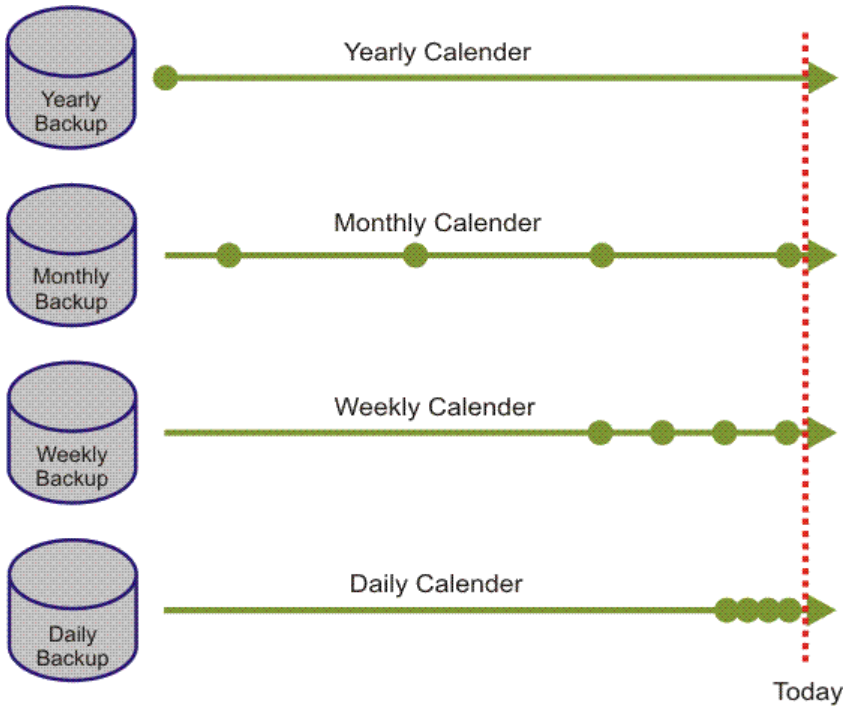
During the ongoing process of calculating the service levels for each of the Metrics, the Engine is often forced to perform a partial calculation for a period that has not yet completed. In order to ensure that it does not need to go right back to the start of the calculation when new data arrives over time, it performs a type of backup of its current 'state' before moving on to its next calculation task. At this point it takes a snapshot of the current variables and values at that point in the calculation and saves this 'state' to the database.

The Business Logic backup process is a mechanism by which the Business Logic code, including the variables' values, is encoded into a binary stream and saved in the database. This mechanism is also required in order to speed up the calculation engine performance in cases of recalculations. The state is backed up from time to time, and is used in recalculation and as an efficiency measure for continuing calculations.

For example, if a recalculation is required for a month retroactively, then instead of recalculating the results from the beginning of the Contract, the closest backed up state before the recalculation date is used and the calculations are performed from that state onwards.

The calculation engine uses pre-defined heuristics to determine when backup is needed and uses the backup capabilities to store the encoded state in the database.

In the figure below, the red dots represent a state backup. The further back the consideration, the fewer the number of backed up states that are taken into account. The logic behind this mechanism is the assumption that recalculation is usually required for the time period closer than one month back.


Yearly Backup
Yearly Calender
Monthly Backup
Monthly Calender
Weekly Backup
Weekly Calender
Daily Backup
Daily Calender
Today

Optimize the System for Recalculation

The service level calculation process requires significant amount of CPU resources, memory and storage. Following is a list of recommendations that can reduce machine load and improve calculation speed.

Note: Some of these recommendations require internal settings not available in the user interface. In such cases, contact CA technical support for more details and instructions.

- **Change the states saving configuration**

  Depending on the Adapter's known delays, you can set the states parameters to better suit your requirements; meaning you can change the number and granularity of the states' save points.

- **Configure the system to calculate only the time units that are really needed**

  Metrics can have up to seven different granularities periods: *year*, *quarter*, *month*, *week*, *day*, *hour*, and tracking period. In some implementations, not all the granularities are necessary. You can turn off unnecessary granularities for uncommitted contracts, via the user interface. Refer to each Metrics granularities tab. For changing a committed contracts Metric granularities, contact CA technical support.

  Note: Avoid calculation of operational periods that are similar to the tracking period.

- **Do not calculate "without correction", "without exceptions" values**

  By default, the calculation engine calculates four different values: *provided*, *provided with corrections*, *provided with exceptions*, and *provided with corrections and exceptions*. You can change this to calculate only the provided value.

  Note: For more information, contact CA technical support.

- **Change the calculation order**

  The PSL calculation order of Metrics can be prioritized to start with your more important Metrics and then to continue to the other Metrics.

  Note: For more information, contact CA technical support.

- **Create more then one PslWriter instance**

  By creating more than one PSL instance (of the calculation engine), you can split the workload and increase the calculation speed.

  Note: For more information, contact CA technical support.

- **Plan implementations to shorten recalculation time**

  To optimize recalculation time, you can:

  - Run the Adapter more often in order to reduce delayed events and avoid large backlogs of data for the Engine to process
  - Turn off unused agents in the Metrics granularity tab.

- Duplicate Metrics and calculate different agents using the same Metric, to balance the calculation load
- Use intermediate Metrics to perform common calculations and share the results with all the other Metrics who require the same data.

- **Plan implementations to reduce the amount of data**

  To optimize the amount of data, use the Adapter to load only aggregated (processed) data. Aggregating the data source information before it is sent to CA Business Service Insight Raw Data Table increases the PSL input reading efficiency.

- **Follow CA Business Service Insight PSL configurations recommendations**

  The PSL Engine can be reconfigured for better performance according to the specific application environment and requirements. Some of the parameters can be set from the user interface and others can be set only from the database system configuration table.

  Note: Consult support recommendations for your PSL configurations.

Logs and Alerts

There are cases where the Business Logic is required to report to the log or trigger an alert message. This is needed when the messages should be sent based on the event processing. Any information that is gathered during the process of calculation and which may be valuable can be sent as an alert. For example, an alert message can be sent when a specific event is under the specified resolution time threshold or in trend analysis when a certain number of consecutive failures has bee reached.

"Out" is a global Business Logic object that enables the formula to send alerts as well as log messages. It has two methods associated with it which have the following form:
`Alert(<Event type>, <Resource name>, <value1, value2>, …<value16>)`

This command sends an alert of a specified event type. However this event type needs to be created manually for the purposes of this alert. The number of values and their type must correspond to the Event type definition.
`Log(<Message>,<Level>)`

This command sends a message to the system log. The first parameter is the information message reported, and can be free text. You can also append the values of variables to this string to give contextual meaning to the message. The 'Level' parameter can take one of the following values:

	Description
	A warning message is reported.
	An error message is reported.
	An information message is reported only when running in the Business Logic Scope. When running in PslWriter, no message is reported. This is the default level. It is used mostly for debugging purposes.

***Example*:**

The following is taken from a case where the Infrastructure information of the event was expected before the actual incident details. An alerting mechanism was set up to notify the Administrator of this condition to prompt fixing the issue.

```
Out.Alert "Site Unknown Alert", Context.ClusterItem, Context.Rule

Out.Log("Fault Event Received for a Site with no infrastructure details: " &
Context.ClusterItem)
```

Breach Root Cause Comments and Event Comments

Breach root cause comment can be set in the Business Logic to explain the service level results. The breach root cause comments are associated with Metrics.

It is also possible to generate Event annotations which are comments associated with the Raw Data Events, rather than with the Metric. Both these types of comments can be viewed from the report data.

Two methods in the Business Logic 'Tools' object allow for the creation of breach root cause and event annotations records:

- **Tools.AddRootCauseComment** (Text, Timestamp, [resourceId])
- Saves a root cause comment. This information can later be useful in generated reports. The saved root cause comment explains a specific situation during execution of the Business Logic formula at a specific moment.The Information parameter specifies the comment should be written.The method receives a timestamp to be saved together with the comment. It also accepts a ResourceId parameter specifying resource related to the method context. (This parameter is optional, and may be omitted.)
- **Tools.AddEventAnnotation** (EventId, Text)
- These methods can be used anywhere within the Business Logic, and the context of where they apply needs to be considered. Adding a root cause comment may be more relevant at the end of a calculation period, when the reason for the service level to be less than expected for that period is known. Lets assume, for example, during the period of a month there were four outages, all caused by a single device. The root cause comment could then be compiled from some of this information about the outages, so that when the reports are viewed for this period it can seen what has contributed towards any service level violation, during this time. The AddRootCauseComment command is best suited to the OnPeriodEnd event handler routine, or other similar function which is run towards the end of the period being calculated.
- Adding an Event Annotation however, is more suited to the actual Raw Data Event processing and favors its usage to the OnXXXEvent (the custom event handler specified in the registration statement) Within this event handler, all of the fields specific to the actual event are available via the eventDetails object.
- An example of how this might look within the Business Logic could be as follows:

```
Sub OnPeriodEnd(Time)
        pctAvailable = (TotalTime-OutageTime) / TotalTime
        Tools.AddRootCauseComment “Violations caused by the
   following items: ” & ViolationCollection, Time
End Sub
…
…
Sub OnIncidentEvent(eventDetails)
    OutageTime = OutageTime + eventDetails(“TimeToResolve”)
```

```
    If eventDetails("TimeToResolve") > 6 Then
        ViolationCollection = ViolationCollection &
    eventDetails("HardwareId")
        Tools.AddEventAnnotation eventDetails.EventId, "Incident " _
          eventDetails("IncidentId") & " not resolved within target
    time 6 hours"
    End If
End Sub
```

Separating Exceptions from Timeslots

CA Business Service Insight Business Logic does not receive exception events. What it does receive is an OnTimeslotExit when an exception period starts and OnTimeslotEnter when an exception period ends. The Business Logic therefore cannot distinguish between exception times and times out of Timeslot. Furthermore, it cannot distinguish between exception types. As a result, it is not possible to implement different logic for exception time behavior and for 'out-of-timeslot' behavior.

One way to implement special exceptions (that is, an exception that does not behave as an 'out-of-Timeslot' period) is by defining dedicated event types, instead of using CA Business Service Insight's built-in mechanism for handling exceptions. These events are generated by reading them from a dedicated data source, using an Adapter.

An Excel spreadsheet (or any other data source) can store these exceptions, and an Adapter can then load the data and generate a response: Exception Enter and Exception Exit events. Alternatively, the exceptions may be added by using Corrections. In addition to the correction, a dummy resource should be defined and associated with these events for the purposes of registration. This resource does not serve a purpose other than a placeholder as it is required by the command.

In order to be able to handle the exception times reported by these dedicated events, the Business Logic formula should register with these Exception events in addition to the normally required Registration for Raw Data Events to use in the calculation.

It is recommended that the Business Logic Expert include a field for exception type in the event type, allowing for the different handling of various types of special exceptions.

This approach has the following characteristics:

- It separates two exception sets-the standard and the special.
- Special exceptions do not have a User Interface for maintenance.
- Reports based on the exceptions generated as Events by an Adapter do not comment on their existence. In cases where the correction mechanism was used, there is a comment. The use of the correction method is recommended to maintain the integrity of the results produced by the system.
- No "With/Without Exceptions" specification considers those exceptions.
- Where the correction mechanism is used, the With/Without correction applies.

Once implemented it is recommended that the Business Logic Expert applies the logic to all of the system Metrics.

There is another method of applying an exception on a single resource if required. This method involves using the resources 'Effective' status. Setting a resource's status to 'Not Effective' means that during this period the Calculation Engine will ignore all Raw Data events which are sent for that resource. By setting a period of time that the resource is not effective by creating new versions of the resource, one at the start of the exception period, and another at the end of the exception period.

However, if the resource is part of a Clustered Metric and the resource is becoming effective and not-effective in the same calculation period, only the last period where the resource was effective will be taken into consideration in the result as stated above. In that case it is advisable to use the custom attributes feature. An additional attribute for the resource stating the status of the resource can be managed and the business logic formula will query for the status of the resource in each relevant place in the script.

Memory and Performance Considerations

Following are a set of situations that should be considered when designing Business Logic solutions. The situations described are cases in which the performance of the calculation Engine may be negatively impacted:

- **Parameters**

  If a value of a parameter is required in the code, the creation of a global variable to assign the parameter value to it is recommended. In addition, whenever the value of the parameter is required, use the global variable instead. This prevents a situation where the Engine creates the parameters map for each parameter call.

- **Use of maps in Clustered Metrics**

  Large global map objects in the Business Logic for Clustered Metrics should be used only with careful consideration. While the Engine is calculating a Clustered Metric it is busy loading the global variables from the previous states for each item in the cluster separately.

- **Business Logic registration**

  Filtering the Raw Data Events solely by the registration methods is recommended. Adding internal filtering using an "if" statement within the code, will result in additional processing time. More importantly, additional overhead is required by the Engine for retrieval and processing of raw data records that are not required.

- **Avoiding the use of Dispatcher.RegisterByEventType**

  Improves performance. Using this registration method means that you are registering to all resources in the system and not only to resources which have events of that specific type. Thus every change in resource impacts the Metric calculations. Another disadvantage of using this registration method comes at Metric run time when it accesses the raw data. It then needs to filter out from the raw data only the events with the specific event type and ignore the other events.

- **Dispatcher.Register**

  When you are using Dispatcher.Register, *always* verify that you specify the 3rd parameter. Registering without the 3rd parameter is exactly like doing registrations by event type (Dispatcher.RegisterByEventType). In other words, make sure that you use at least one other parameter beside the first two.

- **Calculation granularity**

  It is important to turn on only the agents that are required for calculation and drill down purposes. Calculation of all of the agent's time units is very processor intensive.

How to Activate the Contracts (Contract Manager)

The activation of a Contract is done by committing it. (For further details, see the Online Help.)

The activation of the Contract triggers the Engine to initiate the calculations for the Contract's Metrics and to start producing results for the Contract.

Before activating the Contract check whether all of the following conditions have been fulfilled:

- All the Metrics must have Business Logic defined for them (In the Metric or as a linked module) which has been tested and contains no errors.
- All the Metrics have a defined dashboard threshold. (For further details, see the Online Help.) It is important that the thresholds are already defined so that during the process of calculation the Dashboard will be able to evaluate the limits for the Metric.
- The effective dates are in accordance with the correct time period and that there are available raw data records. Check also that the results are as expected using the business logic scope.

Once the Contract has been committed check whether the activation has commenced successfully and that calculations are progressing as expected.

Follow these steps:

1. Check that all of the CA Business Service Insight service components have been started (especially the calculation engine, which includes both the PSLWriter and the PenaltyWriter). It is recommended that all service components are running whenever calculation is required.
2. Diagnose the Contract-the Contract diagnostic feature displays the results of a series of tests performed on all of the Contract's Metrics (and Penalty formulas if they are used). The severity of the test result is provided along with a suggested resolution procedure. It is recommended that a diagnosis is performed whenever activating a Contract, as well as after the calculation for that Contract has completed.
3. Generate the 'Calculation Status' free-form report. This report is bundled as part of the initial CA Business Service Insight installation, and is located in the 'Admin Reports' bundle folder. It provides information on the calculation progress and can be used at this point to verify whether the PSL engine is progressing and whether calculation has completed. Review this report to evaluate whether there are any potential problems in the calculations.

The report has the following column fields:

Field	Description

Contract	Holds the name of the Contract. The list contains the names of the effective and not effective Contracts.
Metric	holds the name of the Metric within the Contract. The list contains all of the Metrics contained within each Contract.
Tracking Period	holds the Metric calculation period. The list will display an entry for each calculation time unit of the Metric according to the agents that are active and based on the granularity definition of the Metric. In cases where the calculation period is the tracking period, this will be noted.
Updated Up To	holds the last result update time. This indicates that the result for the specific Metric is available up until this date. For example, if 01/01/2006 is shown, it indicates that all of the results for that Metric in that time unit are updated up until this date and that reports are available for it up until this date.
Last Cycle Begin At	holds the time the cycle when the calculation in which the result of the Metric was updated began.
Requires Recalculation From	in cases the engine tags a certain Metric for recalculation and it has still not been updated, the resulting date will appear here specifying the time from which the results are not relevant and therefore require updating. This can occur in any cases of recalculation.
Last Update At	the time at which the Engine updated the record with the last result.
Handled by Engine #	holds the number of the Engine assigned to deal with handling the calculation of the specific Metric.

This report can also provide the following information based on the available raw data:

- The time it took the Engine to calculate a single Metric. It is possible to see, by sorting all of the Metrics that were calculated in a single cycle by their updated time, how long it took the engine to update each Metric. All of the records with the same "Last Cycle Begin At" value are calculated during the same cycle and the update time is the "Last Update At" time. It is possible to evaluate the time it took the Engine to calculate the entire Metric with its underlying agent's time units as well as each of the time units.
- The time it took the Engine to calculate a complete Contract. This is done by examining the update time of the first Metric of the Contract and the last update time of the last Metric of that Contract, and calculating the time in between.

Full Recalculation of Contract Metrics

The process of recalculation in the current context is the initiation of a full system recalculation by either the Data Source Expert or the Business Logic Expert. It is not the Engine recalculation performed during a regular calculation process. This type of action is usually performed during or after the process of the Contract configuration when various malfunctions may have been identified.

It is recommended that a full recalculation is commenced only once the system is in a stable state (that is, not during building of the system) before going live on site with it.

Recalculation is currently done by executing a SQL script on the database. This script cleans the PSL table and all of the associated supporting tables that are a part of the calculation process. This script should be approved and validated by the CA support team before being executed, as occasional structure changes to the system could have resulted in changes to the database schema and/or dependant objects.

Note: Before running the script, all service components must be stopped and after the execution, the service components may be restarted in order to restart the calculations again.

The following situations may signify the need for a full recalculation:

- Problems with the definitions in the Contract-if at this stage, it is found that a Metric target was incorrectly defined or a Metric threshold is wrong.
- Errors in the Contract.
- A requirement to re-evaluate Dashboard thresholds or regenerate SLA alerts.

Note: Contact CA Support to discuss this option if required and also to obtain copies of the relevant scripts for the version of CA Business Service Insight that is installed.

Create Deliverables (Contract Manager)

At this stage all of the outputs of the system are configured to meet the requirements, including the creation and formatting of all reports.

The deliverables configuration includes the following:

- Defining the security settings (user permissions and groups).
- Creating saved reports.
- Creating booklets.
- Creating Contract export templates.
- Creating Service Delivery Navigator (SDN) views
- Configuring Dashboard maps
- Creating service level alert profiles.

Define Security Settings (Administrator)

When defining security settings, the CA users and their associated permissions need to be configured. These settings include defining what information should be accessible to a user (which entities within the system can the user view or edit). These permissions can be set at various levels from user groups, roles or even individually per user. Accessible information is defined in relation to Contract parties and can be set directly to the user, or inherited from the user group to which the user belongs.

It is at this stage that the main roles are configured and the user groups associated with them, so that when a new user is added, they need only be attached to a group in order to inherit the relevant settings.

Permitted actions are configured in the roles, and are issued to the user by directly associating the user with a role, or by the user group to which the user belongs. Permitted Dashboard related actions are also defined in the attached role.

It is recommended that the Administrator determines which user groups and roles should be defined, and their required permission in order to be able to set up a structure that allows for the easy addition of users.

How to Create Reports

Use the following process to create a report:

1. Create all of the report's required folders-all of the folders should be created in advance in order to be available when saving each new saved report. Usually each Contract has its own folder including an executive folder for the high level reports.
2. Define the report filtering criteria using the Report Wizard-each saved report creation starts by generating the report using the Report Wizard. In the Report Wizard, the required filtering criteria are selected and the report is generated. IT administrators can set report parameters designating user-defined fields to be filled in by the report user/viewer, enabling generation of report results specific to that user's interest.

   When generating a report for the purposes of setting it as a saved report, it is very important to define the filtering criteria as flexibly as possible. This prevents unnecessary work as the system grows and evolves. The aim should be to ensure each report always displays current and updated information. For example, if a report currently shows three service components, in the future, when adding new service components, it is important that this service is added automatically to the report and does not require a new report setup. Or, if reporting by month and the report currently shows three months, then in the next month it should show four months including the last month. In many cases the reports should always show the last 6 months worth of data. These types of 'sliding-window' reports are extremely useful as opposed to fixed duration reports since they require no additional attention once created.

The following are a few tips for setting the filtering criteria for saved reports:

Criteria tab

- Using the 'Business Data Only' option limits the displayed data only to that which is related to the tracking period of the Metric.
  - Always give preference to the *group by option* or *report by option* rather than selecting the specific Metrics to be reported on
  - Set a dynamic time range. If this is set as a fixed time range the report will always show the same results. (i.e. Last 3 Months is a dynamic range)

Misc tab

- Check whether or not the "incomplete periods" should be presented in the report. If they should be, then select this option from the list. Usually when configuring reports, the incomplete period is excluded because it's not a final result and therefore not a business result.

- Design the report layout-once the report has been generated, it is possible to design its layout using the Design Tool in the Report Page. The design may have specific requirements depending on the intended recipient of the report. It is recommended that a number of design layouts are created, one for each possible scenario and these designs applied to the remainder of the reports to be generated. Hence, when defining the criteria of the report, in the *Display* tab, choose *Design from*.

  Note: For tips on how to use the editing tool to design the layout of reports see the next section.

General tab

- Save the report-once the report has been generated and designed, it can then be saved in the relevant folder.
- During the process of saving the report, it is associated with users who have access to it. It is therefore important to have a user group already defined so that it is possible to associate reports with users. Users with the correct permissions can be attached to the report at a later stage by using the folder options.
- Associate the related reports so that the navigation between reports which are similar, or have a relationship to each other are simplified for the users of those reports.
- In the Reports Folder page, one can create reports, report groups, compound reports, free-form reports, booklets, shortcuts and folders as well as search for a report(s).

  From the Reports menu, click Report Folders. The Reports Folder page is displayed, showing a list of saved reports.

Deviation Report

The Deviation value is calculated automatically by the CA Business Service Insight Engine for Metrics that have a target. The value represents the difference between the actual service level and the target. The deviation calculation formula, calculated automatically by CA, changes according to the service level definition: whether the service level target is a maximum (when the actual service level is "no more than"), or a minimum value (when the actual service level is "no less than"'). See below an example of how the formula changes:

Target Statement	Service Level Threshold	Deviation Formula
Service should be available at least 99% of the scheduled time.	Target is the minimum threshold	$Service_{Dev} = (-1)x \frac{Service_{SL} - Service_{SLT}}{Service_{SLT}}$

Average MTTR should not exceed 4 hours per month.	Target is the maximum threshold	$Service_{Dev} = \frac{Service_{SL} - Service_{SLT}}{Service_{SLT}}$

Where $Service_{Dev}$ = Service deviation

Where $Service_{SLT}$ = Expected service performance

Where $Service_{SL}$ = Actual service performance

The following example illustrates a minimum deviation calculation:

Service should be available for at least 99% of the scheduled timeslot. The actual service level is 98% during the scheduled timeslot.

$$Service_{Dev} = (-1)x \frac{98\% - 99\%}{99\%} = 1.0101\%$$

Deviation reports are used for high level views of guarantees of a foreign nature (different type of calculation), and to aggregate them into a single bar with common grounds.

If, for example, in the contract is the following matrix:

Service-Helpdesk	**Service Domain-Ticket management Priority 1**	**Service Domain-Ticket management Priority 2**	**Service Domain-Ticket management Priority 3**
	P1 average resolution time	P2 average resolution time	P3 average resolution time
	P1 % of tickets resolved within T1	P2 % of tickets resolved within T1	P3 % of tickets resolved within T1
	P1 % of tickets responded within T1	P2 % of tickets responded within T1	P3 % of tickets responded within T1
	P1 average response time	P2 average response time	P3 average response time

The results of generating a Service Domain report filtered by Helpdesk Service are as shown in the following graph.

The above report allows the Service Level Manager to see how well the helpdesk is performing based on various priorities, regardless of the contract or type of obligation.

All helpdesk Metrics are aggregated into a single bar based on their priority.

For example, the Priority 1 bar shows the three Metrics defined within the Metric and aggregates their deviation into a single value. (The aggregation method can be chosen in the report wizard to be average / minimum / maximum.

With such a report, the manager can conclude for example, that he/she needs to invest more resources in Priority 1 tickets or change the contracts related to them.

This example shows that modeling provides both the report on a single obligation which shows whether a contract was honored or breached, but also a broader management report that allows the Service Level Manager to manage his resources more efficiently and thereby improve his service components.

Free-form Reports

Free-form reports enable users to generate reports based on SQL queries of the CA Business Service Insight database or from any other external data source which can be accessed via a connection from the CA Business Service Insight server. This also includes any other types of data source which can be accessed via ODBC, such as Excel, Access, Lotus Notes, Text files etc. Free-form reports are commonly used to configure statistical reports based on data created by the Business Logic commands Tools.SaveRecord and Tools.SaveFields.

Free-form reports connect via a connection string to a selected database and execute an SQL query on the database using a query string. Parameters can be added to both strings in order to create dynamic reports that enable the user to enter or select specific values to include in the query, such as the user name and password for connecting to the database.

Free-form reports are displayed in Chart, Data and Filter tabs similar to reports generated using the Report Wizard.

Note: Free-form reports can include charts only if all columns, apart from the first column, are numeric. The data in the first column is used for the X-axis titles. Column names are used for other titles.

Due to the fact that free-form reports use direct access to a database and an open SQL query, maintenance is problematic. Great care should be taken not to affect the underlying data which serves as a source for the free-form reports. When reports are generated from an external data source, it is recommended that a notification process be set up to ensure that those data sources are not changed without first consulting the Contract Manager responsible for the free-form data reports.

General Information to keep in Mind when Creating Free-form Reports.

- Due to international date formatting problems, it is often useful to specify the exact format for the date by creating a custom formula in the Business Logic. This converts the date to a string with the same format every time and avoid issues of US vs EU date formatting. It should be used for dates being written out to the table T_SLALOM_OUTPUTS via the Tools.SaveFields or Tools.SaveRecord commands. The sample formula is provided below:

Function FormatDate(DateField)

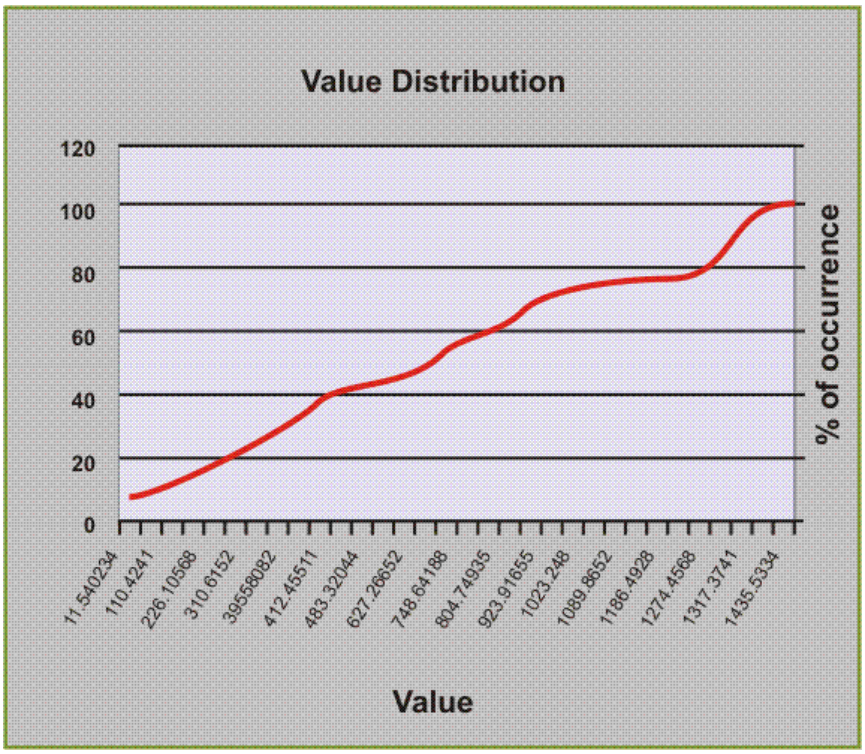
```
    If DateField = "" Then
      FormatDate = ""
    Else
      Dim PeriodYear, PeriodMonth, PeriodDay, PeriodHour,
  PeriodMinute, Periodsecond
      PeriodYear    = DatePart("yyyy",DateField)
      PeriodMonth   = DatePart("m",DateField)
      PeriodDay     = DatePart("d",DateField)
      PeriodHour    = DatePart("h",DateField)
```

```
      PeriodMinute  = DatePart("n",DateField)
      Periodsecond  = DatePart("s",DateField)
      FormatDate  = PeriodDay&"/"&PeriodMonth&"/"&PeriodYear& _
 " "&PeriodHour&":"&PeriodMinute&":"&Periodsecond
    End If
End Function
```

- When using the 'time' parameter from the Business Logic event handlers, or any timestamps from the eventDetails object, always be aware of the impact of the timezone shift. Be aware that the dates when written to the table may be in UTC time rather than local time. You may need to use the Tools.GetLocaleTime() function to correct this.
- You can still use the Report Designer utility (when a free-form report produces a graph) to customize the appearance.
- The PDF exporting options for the free-form report are customizable using the Report Parameters section of the free-form setup screen. Things such as page layout (landscape, portrait)
- HTML can be embedded in the reports to perform different functionalities, such as adding hyperlinks or changing column and row colors, fonts or display settings.
- Database Views (Oracle functionality) can be created on the T_SLALOM_OUTPUTS table in order to simplify the SQL required for the reports.
- When specifying parameters for the reports, it is possible for them to be set in a number of ways including: Free Text, Numeric (validate), Password (characters hidden - useful for passwords to DB connections), Date (using the provided date picker utility) and Lists (created by specifying an SQL statement to populate the list)
- Parameter values specified in the SQL section of the free-form report need to be substituted with the parameter name, preceded by the '@' symbol. i.e. @PARAM1. The use of quotes around this parameter value may be required if the values are strings.

Generic Histogram Free-form Reports

The following query may be used in a free-form report in order to present the distribution of values in a table by percentage, as displayed in the following chart:



It is possible to see in the above chart what proportion (percentage-wise) of the values are below 11.5 (0%), below 804.74 (~50%), and below 1435.53 (100%).

If the SLA specifies targets such as; "x% of the values should be below y", the results of this free-form assists in finding the x, y values that ensure compliance with the SLA.

The following parameters are used in the query:

- **@Query**-a select statement with the following form; select *some_value* **val** from *some_table*. The alias val is obligatory. This query provides the values for which the distribution is analyzed.
- **@Buckets**-the number of values on the x axis. Source values are rounded to these numbers. For example, if you specify @Buckets = 100, then values of the source data is divided into 100 groups and the query will calculate how many values fell within each group. The higher @Buckets is, the more accurate is the result, but the chart is not visually appealing.
- **@Relation**-the direction of accumulation. Possible values; "<=" (when target is no less than) or ">=" (when otherwise).

The query may be run against the data source or T_SLALOM_OUTPUTS for the best results.

The following query produces the graph as shown above:

```
select val,100*records/(select count(*) from (@Query))
from
(
  select x.bucket_val val,
    sum(y.records) records
  from
  (
    select round(val/bucket_size,0)*bucket_size bucket_val,
      count(*) records
    from
    (
      select (max(val)-min(val))/@Buckets bucket_size
      from
      (
        @Query
      )
    ) params,
    (
           @Query
       ) source
    group by round(val/bucket_size,0)*bucket_size
    order by round(val/bucket_size,0)*bucket_size
  ) x,
  (
    select round(val/bucket_size,0)*bucket_size bucket_val,
      count(*) records
```

```
    from
    (
      select (max(val)-min(val))/@Buckets bucket_size
      from
      (
        @Query
         )
    ) params,
    (
            @Query
       ) source
    group by round(val/bucket_size,0)*bucket_size
    order by round(val/bucket_size,0)*bucket_size
  ) y
  where y.bucket_val @Relation x.bucket_val
  group by x.bucket_val
  order by x.bucket_val
)
```

Following is a Sample Parameter list (as XML) which could be used:

```
<custom>
     <connection>
          <params/>
     </connection>
     <query>
          <params>
               <param name="@Query" disp_name="Data Type" type="LIST">
                    <value>PDP Context Activation Success</value>
                    <list>
                         <item>
                              <value>select success_rate as val from
PDP_Context_Activation_Success.CSV</value>
                              <text>PDP Context Activation Success</text>
                         </item>
                         <item>
                              <value>select throughput as val from [gprs throughput
volume by apn.csv]</value>
                              <text>Throughput of a Single APN</text>
                         </item>
                         <item>
                              <value>select throughput as val from [Generic GPRS
Throughput.CSV]</value>
                              <text>Generic Throughput</text>
                         </item>
                    </list>
               </param>
               <param name="@Buckets" disp_name="X Axis Values" type="LIST">
                    <value>100</value>
                    <list>
```

```
                              <item>
                                   <value>25</value>
                                   <text>25</text>
                              </item>
                              <item>
                                   <value>50</value>
                                   <text>50</text>
                              </item>
                              <item>
                                   <value>100</value>
                                   <text>100</text>
                              </item>
                              <item>
                                   <value>250</value>
                                   <text>250</text>
                              </item>
                              <item>
                                   <value>500</value>
                                   <text>500</text>
                              </item>
                              <item>
                                   <value>1000</value>
                                   <text>1000</text>
                              </item>
                         </list>
                    </param>
                    <param name="@Relation" disp_name="Violation of threshold means"
type="LIST">
                         <value>providing too little</value>
                         <list>
                              <item>
                                   <value>&gt;=</value>
                                   <text>providing too little</text>
                              </item>
                              <item>
                                   <value>&lt;=</value>
                                   <text>providing too much</text>
                              </item>
                         </list>
                    </param>
               </params>
     </query>
</custom>
```

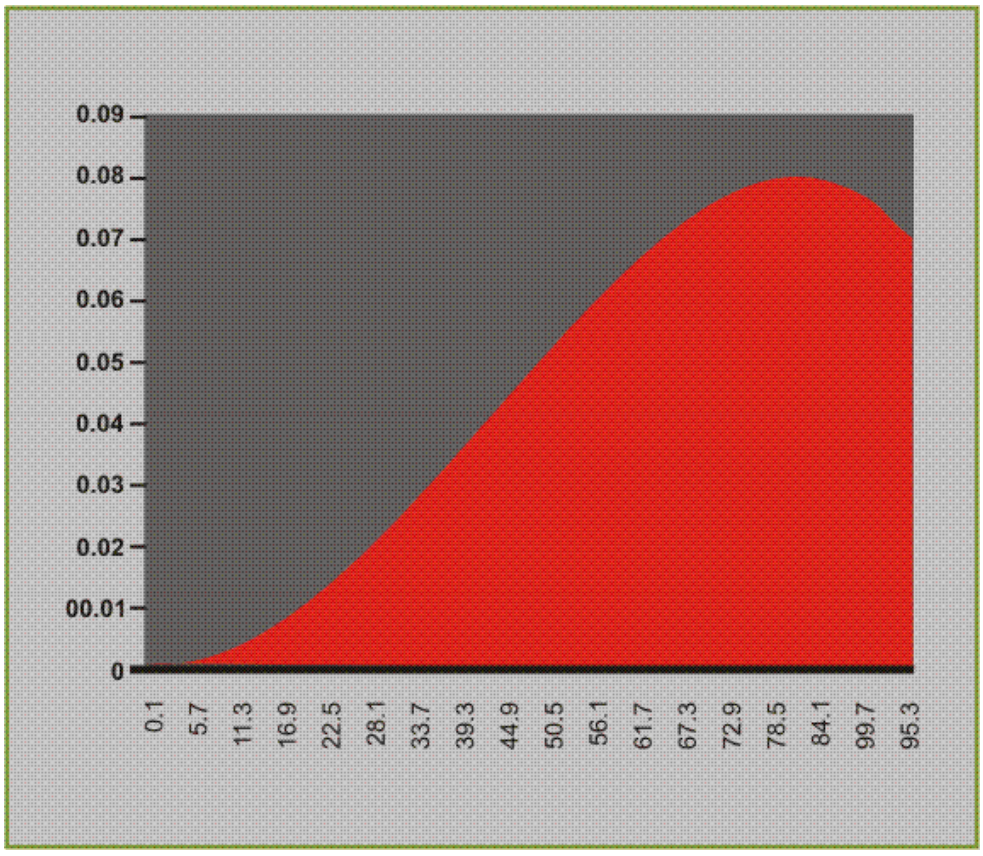
Comments

- The query has been optimized for Oracle and SQL Server. For other ODBC data sources, one may need to add 'as' before column aliases and apply other modifications.

- When exporting results to Excel, it is recommended that the user generate an XY (Scatter) report to represent it. When the chart is dotted it will also be possible to view the scattering of the values.

Generic Normal Distribution (Gaussian) Free-form Report

The query detailed below may be used in a free-form report to present the normal distribution of values in a table, as demonstrated in the following chart:



The following parameters are used in the query:

- @Query-a select statement of the following form:

```
select min (something) min_val,
          max (something) max_val,
          avg (something) avg_val,
          stddev (something) stddev_val
from table_name
```

  x_val is required. This query provides the values for which the distribution is analyzed.

- @Buckets-the number of values on the x axis. Source values are rounded to these numbers. For example, if you specify @Buckets = 100, then values of the source data will be divided into 100 groups, and the query will show the normal distribution value for each group. The higher @Buckets is, the more accurate is the result, but calculation is heavier. 100 is a good choice for @Buckets.

Again, the query may be run against the source data, or T_SLALOM_OUTPUTS for the best results.

The following query will produce the graph as shown above:

```
select (max_val-min_val)/@Buckets*serial,
     (1/stddev_val*sqrt (2*3.14159265359))*exp ( -1/(2*power (stddev_val, 2))*power
(((max_val-min_val)/@Buckets*serial-avg_val), 2))
from     (
          @Query
     ),
     (
          select     rownum serial
          from  t_psl
          where  rownum < @Buckets
     )
order by serial
```

Its corresponding XML parameters:

```
<custom>
     <connection>
          <params/>
          </connection>
     <query>
          <params>
               <param name="@Query" disp_name="Data Type" type="LIST">
                    <value>PDP Context Activation Success</value>
                    <list>
                         <item>
<value>
select min(success_rate) min_val,max(success_rate) max_val,avg(success_rate)
avg_val,stddev(success_rate) stddev_val
       from PDP_Context_Activation_Success.CSV
</value>
```

```
                              <text>PDP Context Activation Success</text>
                         </item>
                         <item>
                              <value>
select min(throughput) min_val,max(throughput) max_val,avg(throughput)
avg_val,stddev(throughput) stddev_val
       from [gprs throughput volume by apn.csv]
</value>
                              <text>Throughput in Kb</text>
                         </item>
                         </list>
               </param>
               <param name="@Buckets" disp_name="X Axis Values" type="LIST">
                    <value>100</value>
                    <list>
                         <item>
                              <value>25</value>
                              <text>25</text>
                         </item>
                         <item>
                              <value>50</value>
                              <text>50</text>
                         </item>
                         <item>
                              <value>100</value>
                              <text>100</text>
                         </item>
                         <item>
                              <value>250</value>
                              <text>250</text>
                         </item>
                         <item>
                              <value>500</value>
                              <text>500</text>
                         </item>
                         <item>
                              <value>1000</value>
                              <text>1000</text>
                         </item>
                    </list>
               </param>
          </params>
     </query>
</custom>
```

Comments

- The query has been optimized for Oracle and SQL Server. For other ODBC data sources, it may be necessary to add 'as' before column aliases and other modifications.
- When exporting results to Excel, it is recommended that the user generates an XY (Scatter) or an area report to represent it.

Configure Dashboard Pages

The following section includes recommendation on the content to be configured for the CA Business Service Insight users. The recommendation is in high level and should take in to account specific customer requirements. The pages are described in the following in the context of a specific user which is described via his role in the organization.

Executive Manager Pages

The assumption is that an executive manager might be interested in high level view across departments countries accounts etc. An executive manager usually is not operational and requires views that provide him information to take decisions in terms of strategy. Therefore contractual status rather then current status might be more relevant for displaying in executive manager maps and aggregative Dashview.

For example, the following Dashviews may be included on the Overview page:

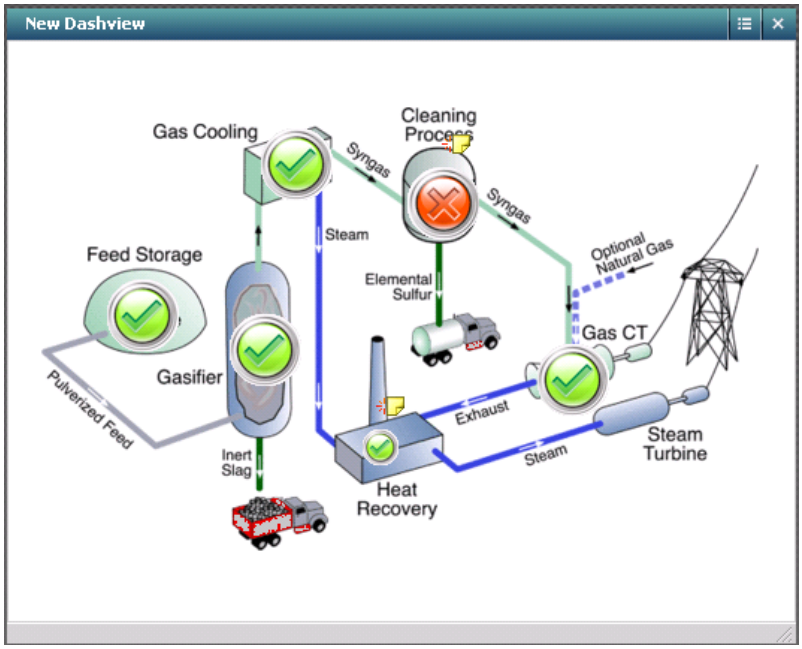
Dashview	Description
Critical accounts	Includes all contract parties that are marked as sensitive. The executive manager selects the contracts or contract parties that from his point of view considered as sensitive.
Overall performance	Includes custom widgets of overall quality, widgets of aggregative view that include all KQI's of the accounts.
Departments	Uses background image with organizational chart and place the widgets on the relevant departments. Usually Service Component Group is useful here depending on what a department represents in the organization.
Geographies	Uses background image with a geographical map and locate contract party groups widgets on the relevant locations.
Financial performance	Includes widgets that include aggregative information on financial metrics
URL	Includes pages from the corporate portal for example leads in the sales force for example

Recommended Reports for the page:

Report	Description
Deviation report	Ten worst contracts for a certain period, provides information on the areas that are in the worst performance in terms of the service delivery.
Financial report for the current month	Summarizes the financial status aggregated thru time

Process page:

This page should contain a Dashview that presents a process diagram with widgets that represents each chain in the process as in the example below:



Contract Manager Pages

Pages configured for a Contract Manager display how well the service is being delivered for each of the accounts he is managing. Dashview of the relevant Contracts in order to update the manager on the current service level provided on the obligations of the Contracts that are under his responsibility. In addition, such a page displays available reports for each of the service components that are included in the Contract.

The Overview Page includes the following dashviews:

Dashview	Description
My accounts	Widgets of all the Contracts or Contract Parties that are under the responsibility of the Contract Manager.
My services	Service components across the accounts managed by the Contract Manager.
Financial performance	Widgets that include aggregative information on financial metrics for the managed accounts.
URL	Portal such as the CRM system

Create an Account Page for each of the managed accounts that provide a view on the specific account obligations. It is recommended to combine views of Current Status calculation aside to Contractual status in order to provide the account manager the ability to be proactive and act fast. For example:

Account Page	Description
Account obligations	Includes Metrics included in the contrac
Financial performance	Widgets that include aggregative information on financial metrics for the managed accounts.

Service Manager Pages

Pages to be configured for a manager in charge of a specific service or domain, which display a detailed view of the service objectives across the different Contracts and the states of its objectives. Furthermore, such a page includes reports that highlight the service critical service-level parameters being measured.

The Dashviews included in the pages of the Service Manager are similar to the Account manager pages, where only the information is displayed and aggregated in a service level rather then in the contract or contract party level.

Home Pages

A dashboard page can be assigned as a home page, so as to enable a customizable gateway for interaction with the system, providing quick access to information and actions.

Add Service Level Alert Profiles

The alert profile allows the definition of conditions under which an alert notification is sent to a predefined recipient using a specified device.

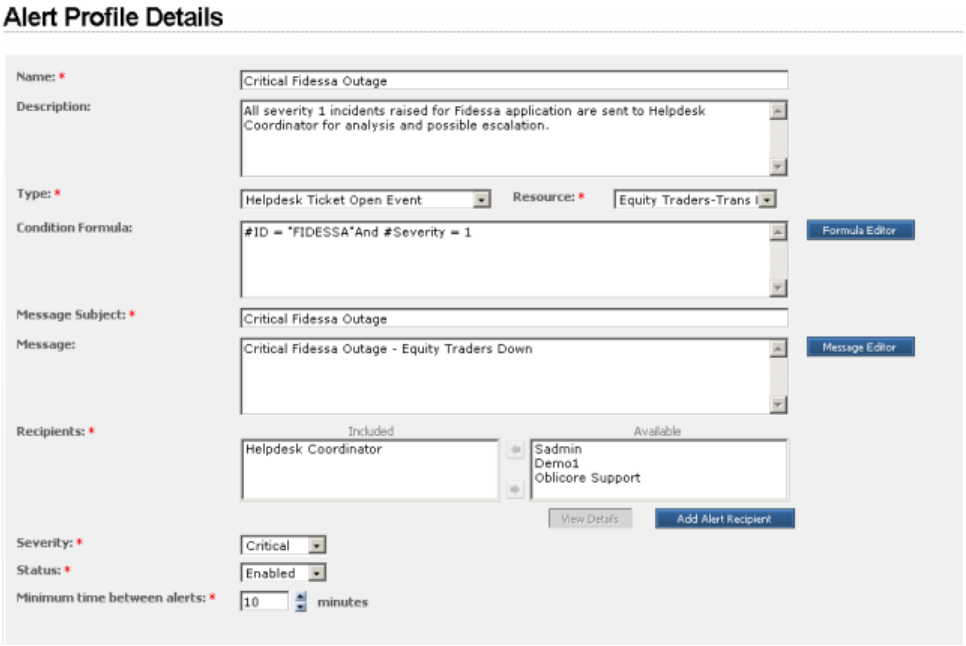
Before creating any profiles, it is worth considering which conditions are important enough to require notification and also who the intended recipients should be. It is important to define profiles which can assist the Contract Manager and System Administrator to effectively manage any problems and service issues encountered. All notifications should be specific in directing the required resources to address the most urgent issues to prevent a situation of breach.

When adding service level alerts there are a number of fields which can be used for determining the status of the Metric and for deciding whether an alert condition should be triggered or not. Any one of the standard Metric details can be used for filtering and condition criteria (such as Contract, Service, Metric names, targets, timeslots, service level results etc) Service Level predictions can also be alerted upon, in which case some additional values such as best/worst case deviation are provided, and service levels can also be estimated. These are extremely useful for deciding whether a service level breach can be recovered from or not, within a given period.

Alerts based upon the System Log and general sections of CA Business Service Insight are other extremely useful features and permit an alert to be sent based upon any action occurring within the system (that is written into the system log). Since almost all actions are logged, this provides a very wide scope for the alert profiles. Common system log alerting profiles consist of:

- Notification of an Adapter running alert for the Data Source Expert.
- Any 'Error' conditions entered in the log, email alert sent to the System Administrator.
- Custom errors created by the Business Logic relevant to specific conditions from the data sources, can generate an email alert to the group responsible for the service within the organization, and the Contract Manager. (i.e. High priority incident has not been resolved within timeframe > send alert to help desk to escalate the issue)
- Repeated invalid login attempts, can be emailed to the System Administrator to determine is someone is trying to hack into the system
- New Translation Entries arrived from the adapter (i.e. New resource entries have been found in a data source, which may need translation in the system to allow the SLA's to be calculated correctly.

Following is the Alert Profile Details window, showing the configuration of a custom alert profile reporting on a custom event type of the 'Helpdesk Ticket Open Events' - If the criteria is matched, then a critical alert is sent to the Helpdesk coordinator to ensure this ticket is managed efficiently. This could be useful for situations where an application is under scrutiny at some time and requires an extra-special level of care.



Following is an example of implementing a Service Level violation alert which is based upon whether the SLA can still be met, considering the current service level and the remaining time left in the business tracking period of the Metric.

Example: Reversible SLA violation alert

This alert is triggered when an objective (Metric) is currently being violated, but since the best case scenario indicates compliance with the objective, the violation can be avoided if a more suitable service level is provided henceforth.

Type: Service level prediction

Condition formula:

#Deviation>0 And #BestCaseDeviation<=0 And
#BestCaseServiceLevel<>#WorstCaseServiceLevel And #ClusterItem = ""

Message:

Note that the objective '#Metric' of the Contract '#Contract' is currently being violated, but the violation is reversible.

While the Service level target is #Target, the following service levels are predicted:

- Service Level by now: #ServiceLevel (Deviation: #Deviation%).
- Worst Scenario Service Level: #WorstCaseServiceLevel (Deviation: #WorstCaseDeviation%).
- Best Scenario Service Level: #BestCaseServiceLevel (Deviation: #BestCaseDeviation%).

Chapter 4: Installation and Deployment

This section contains the following topics:

Introduction

Once the configuration phase has been completed and the system calculations have been verified correctly, the system is ready for deploying into the production environment. This chapter is aimed at covering all of the steps which may be required during this phase. These steps might vary between installations; however the following items should be checked to see if all the preparations have been made for integrating the system into a live production environment.

- Production hardware and software is available and ready for installation.
- Mail server is configured to support external/internal mail sending.
- FTP Agent existence (if required for data file transfers).
- Data Source feeds are available and working

There are two phases within the installation. During the first phase - the installation phase - the System Administrator should install and integrate the CA Business Service Insight components into the production environment. During the second phase, the system functionality is verified and the system should be put in a monitored status to ensure system processes, and UI functionalities are all operating correctly.

During the installation process it may be required to work on the production servers via remote connection tools which should be installed in the application and web servers to allow a complete and safe installation.

Installation of the Oracle database should be performed by an Oracle DBA expert if possible to ensure the configuration is performed correctly and that it conforms to the corporate requirements which may be in place. Alternatively the database creation can be left to the System Administrator, using the Data Creation Utility supplied with the CA Business Service Insight installation. In this case, the installation should be verified by the DBA to ensure that it has been successfully completed. If the system was already configured on a development system and needs to be migrated, then it is necessary to move the database content to this new production database.

Sometimes, once the complete system has been transferred to the new production environment it is advisable to drop all of the calculated and raw data stored in the system and re-load the system from scratch (keeping in place all of the service catalog, resources and translations of course) This is an excellent way of testing the operation of the entire system within the production environment.

To summarize, the following steps are included in this phase:

- Install and connect the system.
- Load the configured system database if required.
- Integrate connections to email, Exchange, SMS, etc.
- Test functionality and tune performance.
- Install, configure and test remote connections.

- Re-initialize the system ready for 'live' operation.
- Activate the data Adapters to start polling the data sources and loading raw data.
- Switch on the CA Business Service Insight Engine and allow the calculations to commence.
- Complete any necessary documentation, such as, system administration, database and other maintenance procedures.
- Ensure all training is provided to users.

Preparations

In order to ensure an efficient installation; it is important for a few things to be prepared in advance:

1. **The database export of the development environment**. This database export should be performed using CA approved scripts that create an extract (dump) file in order to be able to import it back into the software on the production system.

   Note: It is important that the export be performed by a database user that has sufficient privileges and that this same user is accessible when importing into the database. For this purpose either the '**oblicore**' account may be used since this is always created on each system, or alternatively the '**sys**' account. However, ensure that the '**sys**' password is available on the destination database to allow the import process to occur.

2. **Database scripts**- if the DBA wishes to alter the database creation scripts, then they should be verified in advance by a CA DBA. These scripts should be verified and ready in advance to allow a smooth installation. It is common for different organizations to have comments and changes that must be approved by the CA database administrator to ensure that the database configuration complies with local policies.

   - **Server connectivity and accessibility**-both application and web servers should have a remote connection client installed in order to support external accessibility (if physical access is not possible at the installation time, or in the future for support issues). The external accessibility is also important to allow continued monitoring of the system during a settling period. If external remote access is required, this might take longer to organize due to the additional security implications; hence it should be addressed ahead of time.
     Remote connections can be established using any of the following tools:
   - PcAnyWhere
   - Proxy Host / Master
   - Microsoft Terminal Server
   - Remote Desktop
   - VNC
   - Any other remote connection tool supported by the organization.

   Notes:

   1. Microsoft Terminal Server connects to the server by simulating a server login, unlike PcAnyWhere, VNC, Proxy and other tools.
   2. Some Oracle software cannot be installed via Microsoft Terminal Server, and must be installed locally.

3. **Data source accessibility**-the data source should be accessible for manual manipulation and the ODBC connection should be configured to allow the Adapters to connect to the data source.

4. **Server security**-in both application and web server, a user profile with local administrator rights is required. If the database platform is a standard Windows OS, a user profile with local administrator rights is also required. For Unix platforms, the database administrator should have the appropriate privileges to create the database on the server.
5. **Access to Internet**-this allows the **System Administrator** to connect to the internet for any operating system or application updates if required.
6. **A standard users desktop PC, to test the application functionality**. Note that the application requires the use of some ActiveX controls to be installed, which is often locked down by organizational security policies.
7. **Training schedule with relevant staff**-Introductory training that will allow the staff to perform the user acceptance training and to start working with the system

Installation

The installation process is described in detail in the Installation Guide and includes the following activities:

1. **Database installation**:

   Database installation is the responsibility of the Data Source Expert or Oracle DBA, and under special situations should be under the direction of an CA engineer. The database installation includes the following steps:

   - Oracle installation on the database server.
   - Oracle patch installation on the database server (if required - The latest support service releases of Oracle database software should always be used).
   - Oracle client installation on the application/web server.
   - Oracle Client patch installation on application/web server (if required - Always ensure this version matches the server).

2. **CA Business Service Insight Application installation**:

   The application installation is performed by the System Administrator. In cases where the installation was already done in the test environment (during the configuration phase) during the integration and Adapter testing, then only an initialization state or database import is required. Installation in the production environment may then be required.

   The application installation include the following steps:

   - CA Business Service Insight Database Creation.
   - Application server installation.
   - Web server installation.
   - Adapter installation.

Always install the latest CA service release on application/web servers. During the installation of the application and service releases, SQL scripts are included to ensure the database is upgraded to the correct structure for the release. These are all stored in the application installation directory in the event that the database needs updating again at a later date. (For example, if you need to import the database from a backup that was built on a previous service release. In this case you need to run the latest service releases update script to ensure that the structure of the database is in line with the binary components installed as part of that same service release)

Optionally, you can also install additional support tools, such as PLSQL developer or alternative SQL utility to assist in any lower level data manipulation that may be required.

How to Import or Export Between Environments (Data Source Expert)

This situation covers the migration of Adapters that were configured and tested in a development/test environment onto a live or production environment. It is assumed that the production environment has already been installed with a standard CA Business Service Insight installation and the database is present.

The process is comprised of the following:

To export the database and import into the new environment

1. Stop all work on the development system and choose a logical point in the system to perform an export which can be used for the production system. Shutdown all of the CA Business Service Insight service components and CA Business Service Insight COM+ components. Export using the CA Business Service Insight standard system export scripts. (If required contact CA Support)
2. Take the copy of the database extract and place it on the production server ready for importing. Note that the Oracle versions of the development and production databases must match. Import the database using the standard system import scripts (supplied with the export scripts)
3. Once completed, check for any import issues. If none exist ensure you run the latest service release SQL scripts (if required)
4. Run the 'Little Wizard' shortcut to configure the database for the new production system's settings.
5. Start up all the CA Business Service Insight service components and log in to the production system to confirm the system was imported correctly.

To migrate the Adapters

1. Install the Adapter using the AdapterManager utility with the settings similar to the Adapter you are importing (ensure the naming is exactly the same in order for the next step to work correctly
2. Copy the Adapter configuration file from the development system to the new Adapter folder on the production system. Overwrite the default configuration provided. (You should ensure the file is overwritten. Otherwise, it means the naming is not the same, which then causes problems during execution.)
3. Update the Adapter configuration file-the communication to the CA Business Service Insight server and to the data source should be updated to suit the new environment. The OblicoreInterface section should be updated with the correct Adapter port. The DataSourceInterface section should be updated with the correct ConnectionString or file name pattern or path if required.
4. Ensure that all required ODBC DSNs (Data Source Name's) are set up and working on the new machine.
5. Test the Adapter connectivity.
6. Test the Adapter execution.
7. **Translations**-in cases where there are Translation scripts they must be activated. Check that they are in synchronization with the Adapter and that Translations perform as expected. Where manual Translation is used and not previously completed, further Translations should be performed at this stage.
8. **Adapter Scheduling**-Schedule the Adapter to run as expected. If the Adapter has been defined as an application in a Run Once mode, it can be scheduled using the Windows Scheduler. This can be viewed under Control Panel > Scheduled Tasks. See Adapter Running Modes (see page 85) for further details.

Integration - Mail Server Setup (System Administrator)

To enable the notification features of the system, it is necessary to know which mail server and mailbox is used to send CA Business Service Insight emails. This mail server must allow mail to be relayed through it, since it will be sent as SMTP from the CA Business Service Insight servers to this mail server, using the specified account. After you complete the mail server setup, you can use email features in the Alerts, and Contract Approval features in CA Business Service Insight

Click the Administrator menu and select Site Settings, Alerts. Configure the email definitions in the alert settings section, including Email server, sending address and Sender Name, along with SMS provider information for using SMS Gateways.

Part of the integration testing involves verifying that the application server can reach the organization's mail server, and use this to send CA Business Service Insight alerts.

To test the connectivity between the mail server and the application server, run the following command on the application server:

```
C:\Documents and Settings>telnet ORGANIZATION-MAIL 25
```

ORGANIZATION-MAIL

Defines the mail server that you defined on your email client. Contact your System Administrator in order to get this setting if you are unsure.

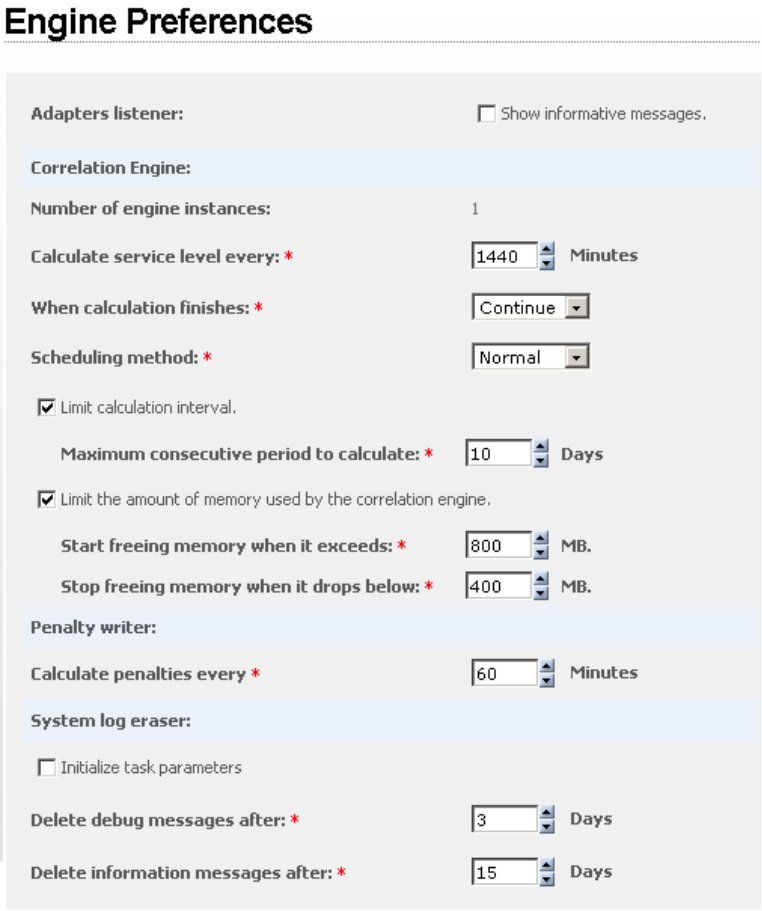
To check the Outlook client mail server:

1. In Microsoft Outlook go to Tools > E-mail accounts and select View or change existing email accounts.
2. Click Change.
3. Copy the Microsoft Exchange server. This server is the mail server of your organization**.**

   If there is a response from the server with this command then the connection has been successful. The response should be similar to the following:

   If you receive any other message, it means there is no connectivity between the two servers. Contact the System Administrator.

Set System Preferences (System Administrator)



The process of setting the system preferences includes applying the relevant values to the system variables. From the Administration menu click Site Settings, Engines to open the Engine Preferences dialog.

For details regarding the various parameter recommendations, refer to the Administrator Guide.

Security Settings (System Administrator)

The security settings include the creation of users, user groups, and roles. By default, all users are associated with the organization specified during the Application installation process. However, it is also possible to create additional organizations if required.

Most of the required definitions have already been completed during the configuration phase, therefore, typically only some fine-tuning is required to define additional settings that may have been identified since that time.

For more details about security setting, refer to the Online Help or to the Administrator Guide.

Specify Settings for SSA Synchronization

When you use CA Spectrum Service Assurance (SSA) to discover services for CA Business Service Insight, you can configure settings to enable automated synchronization. Automation features enable you to keep the list of services up to date, and the service data current.

Note: You must have access to the restful API in SSA to edit these settings.

Follow these steps:

1. On the Administrator menu, click Site Settings, SSA Settings.

   The SSA Settings dialog opens.

2. Enter the following information in the SSA User Authentication area:

   Server URL

   Specifies the URL for the target SSA server.

   User ID

   Specifies the User ID for the SSA server.

   Password

   Specifies the password for the SSA server User ID.

3. Enter the following information in the Synchronization Options area:

   Synchronize Automatically

   Specifies that you want synchronization to occur automatically, according to the synchronization frequency (see next paragraph).

   Set Synchronization Frequency

   Specifies the frequency to search for new services. You can indicate a value in hours or days.

   Manual Sync

   Enables doing manual synchronization from the dialog.

4. Enter the following information in the Default Data Options area:

   Default Services To:

   Allows you to set the default to managed or unmanaged for discovered services.

   Enable calculation of comparison metrics

   Enables setting the default action for setting comparison metrics for SSA services.

5. Click OK.

   Your SSA settings are activated.

Appendix A: Sample Service Domains and Domain Categories

The following table is a compilation of common service domains and domain categories which are commonly used.

Service Domain	Domain Category	Comments
System Availability	% of time available	
	Number of outages/downtimes	
	Mean time to repair	
	Mean time between failures	
Service Availability	Minutes of downtime	
	Number of disrupted days	
Financial Management	Service cost	
Process Performance	% Processes completed on time	
	Number completed process cycles	
Incident Management	% Incidents resolved < T1	
	% Incidents responded to < T1	
	Number of incidents	
	% Incidents converted to problem	
Customer Satisfaction	% Customer satisfaction	
	Average CSI score	
Helpdesk Performance	% Abandoned calls	
	Average call pickup time	
	% FLLR	(First Level Resolution Rate)
Data Quality	% Accuracy	
	% Timeliness	
	Number of errors/defects	

Appendix B: Case Study Examples

This section contains the following topics:

Contract Modeling Examples

For each of the following case studies, use the following categorization items for the described objectives:

- Service
- Service Domain
- Domain Category
- Timeslot
- Target
- Tracking Period
- Unit of measurement
- Calculation Outline
- Contract\Metric Parameters

Case Study 1: System Availability

Consider the following contractual guarantee:

"System X must be available for at least 99.6% of the month during business hours."

This could be described using the following CA Business Service Insight entities:

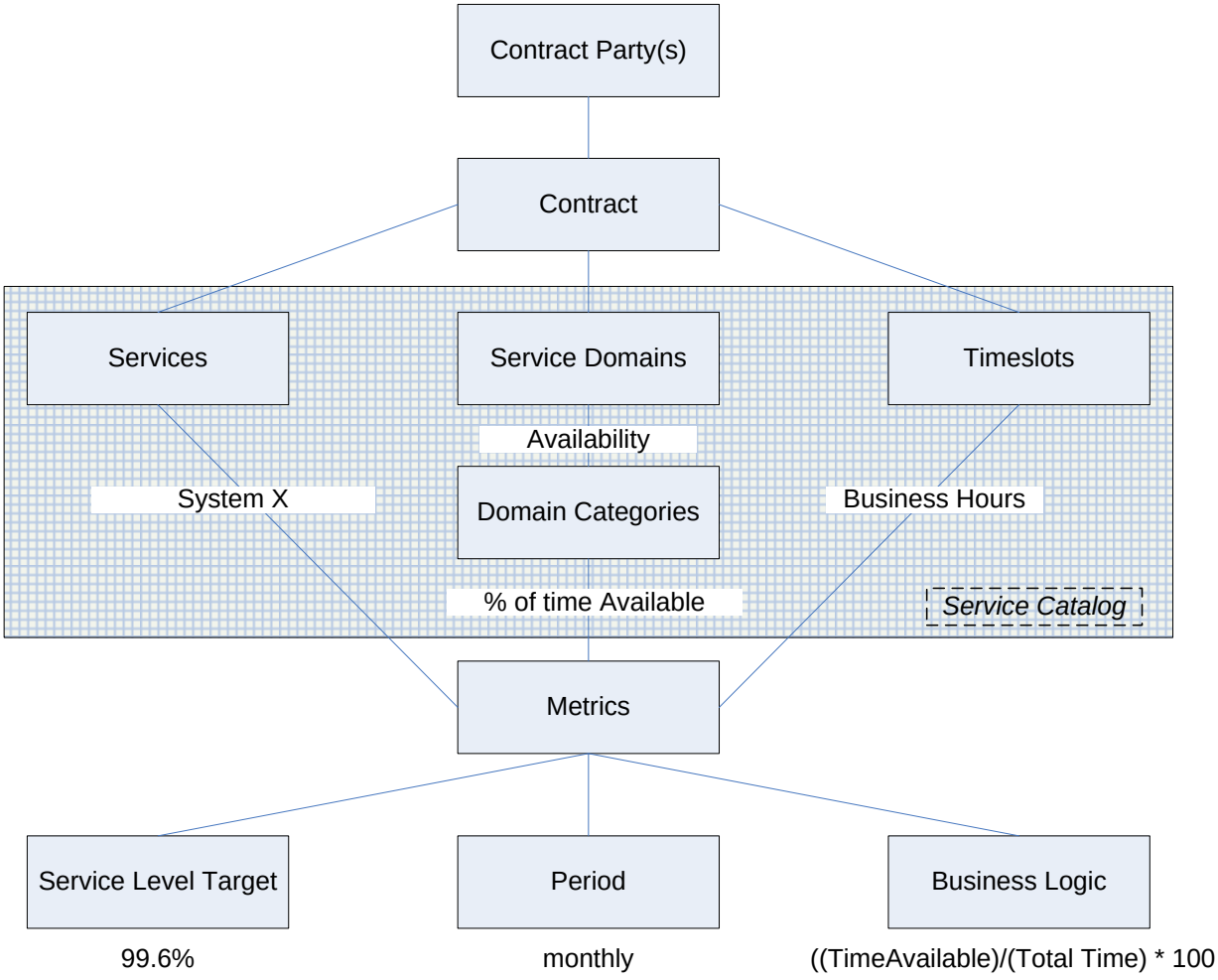
- **Metric Name**: System X % of time available
- **Tracking Period**: 1 month
- **Timeslot**: Business Hours (needs defining further later)
- **Business Logic**: ((time available)/(total time))*100

  Note: This case study is concerned only where the monitoring falls within business hours (which is the time slot of the Metric)
- **Target**: 99.6%

In addition to the previous Metric information, the following items from the system service catalog can also be deduced from the above description:

- **Service**: System X.
- **Service Domain**: Availability.
- **Domain Category**: % of time available.
- **Service Group**: Any group that identifies more than one system for which there can be monitoring. At this stage, it is difficult to establish whether a suitable group can be created.

So, now you can reproduce the diagram from the Contract Modeling section of this document showing these entities in a diagram.



Case Study 2: System Availability 2

The CNP System availability should maintain the following levels:

Environment	Weekdays	Weekends
Production	99.9%	98.9%
Development	90%	NA
Testing/QA	No guarantee	NA
Network	99.9%	NA

Suggested Solutions:

Metric	**CNP System Production Average During Weekdays**
Target	99.9
Tracking period	1 month
Unit of measurement	%
Service	CNP System production
Service Domain	Availability
Domain category	Application availability
Timeslot	weekdays
Metric	**CNP System Production Average During Weekends**
Target	98.9
Tracking period	1 month
Unit of measurement	%
Service	CNP System production
Service Domain	Availability
Domain category	Application availability
Timeslot	weekends
Metric	**CNP System Development Average During Weekdays**
Target	90
Tracking period	1 month
Unit of measurement	%
Service	CNP System development
Service Domain	Availability
Domain category	Application availability
Timeslot	weekdays
Metric	**CNP System Testing/QA Average During Weekdays**
Target	none
Tracking period	1 month
Unit of measurement	%

Service	CNP System testing Q/A
Service Domain	Availability
Domain category	Application availability
Timeslot	weekdays
Metric	**CNP System Network Availability**
Target	99.9
Tracking period	1 month
Unit of measurement	%
Service	CNP System network
Service Domain	Availability
Domain category	Network availability
Timeslot	Always

Case Study 3: System Response Time

The following case study illustrates response time metrics. A Contract can be modeled in several ways, each with its advantages.

The following example examines various modeling methods:

Suggested modeling Solution A	**Maximum Value**
CNP system transaction response time	Cannot exceed 750 milliseconds per month
Metric name	Maximum transaction response time
Target	750
Tracking period	1 month
Unit of measurement	Milliseconds
Timeslot	Always
Service	CNP System
Service Domain	Performance
Domain Category	Maximum transaction response time

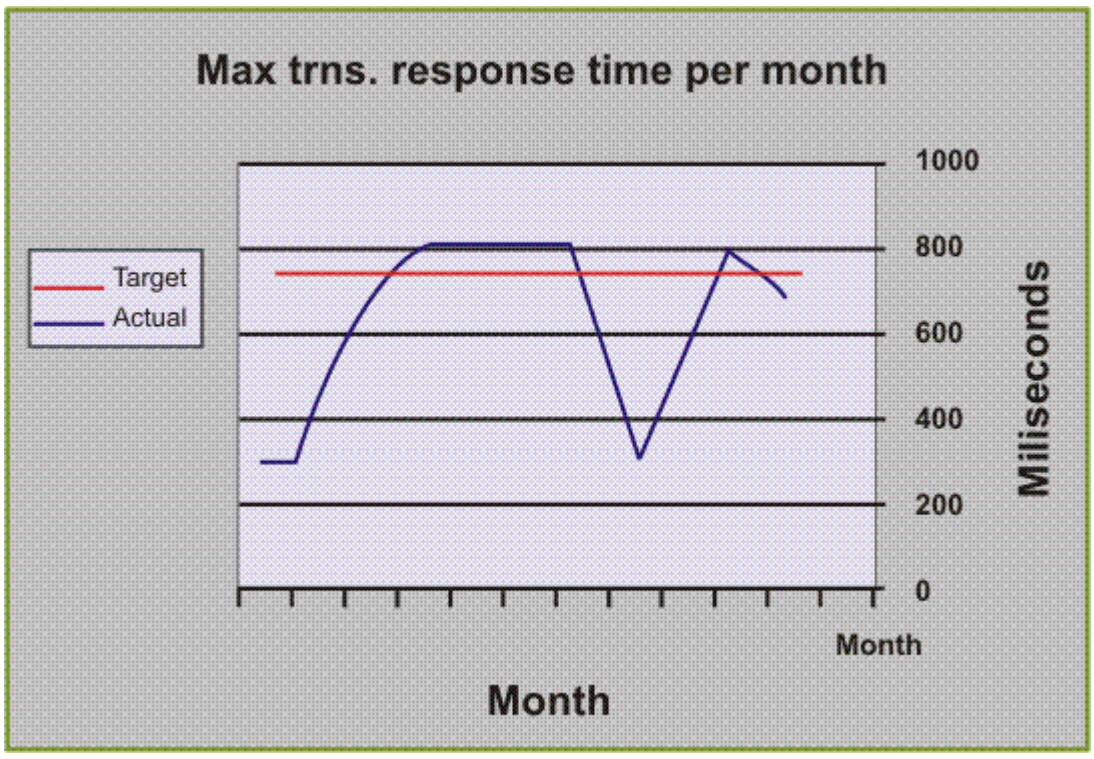
Based on the above matrix, how is the actual service level calculated?

Based on the Domain category definition, it seems that the actual service level is calculated as a maximum value. This implies that for all transactions performed during a month, the transaction with the maximum value is captured and this value is compared to the target.

Note: The service level calculation is based on an aggregation of raw data over a given time period. For each time period, the Metric provides a single result. The target of a Metric is not compared to a single transaction, but is compared to a monthly result which is a periodic aggregation of all transactions within that period. The Contract Manager must make sure this result reflects the contract on the one hand, and the quality of service on the other.

Note that measuring the response time as a maximum value is a very strict obligation and very difficult to achieve in practice. Measuring a maximum level, means that a single transaction of 751ms out of a million transactions conducted during the course of a month is enough to cause a breach of the contract. All bars in the reports will therefore be red and will not reflect the real quality of the service that has been provided.

The following figure depicts a typical report under these circumstances.



Any transaction that exceeds the target will be regarded as a breach of contract, but as a base for understanding the actual quality of service provided it is very poor, as it only reflects a single transaction and nothing is known regarding the rest of the transactions, such as, was it a single failure or a trend? If it was not an isolated incident, then how many failures were there, or what is the ratio of failed transactions to the total number of transactions performed during the month? There may be a number of months with such occurrences and hence a breach of contract, but what is the trend? Is it improving or getting worse? All of these are questions that the Service Level Manager might ask and the report should be able to provide the answers.

Note: When defining the Metric and its associated calculation outline, it is very important to envisage how the results will appear in a report. This report must provide two crucial items:

- Has there been a breach of contract?
- It must provide managers with enough data transparency and ability to investigate the root cause of the failure and also give managers the necessary tools to fully understand their service components.

Suggested modeling Solution B	**Average response time**
CNP system transaction response time	Must be no longer than 750 milliseconds per month
Metric	Average transaction response time
Target	750
Tracking period	1 month
Unit of measurement	milliseconds
Domain Category	Average transaction response time

Calculating the average response time gives a better idea of the monthly quality of service, and yet at the same time is still able to reflect those months with extreme or out-of-contract response times.

Suggested modeling Solution C	**Percentage of transactions that were successfully completed under the threshold.**
CNP System Transaction response time	Must be no longer than 750 milliseconds per month
Metric	Successful transaction response time
Target	100
Tracking period	1 month
Unit of measurement	% of success

Metric parameter	750 ms
Service	CNP System
Service Domain	Performance
Domain Category	Successful transaction response time
Timeslot	Always

Using this method, the calculation will be the percentage of transactions that were successfully completed under the threshold of 750 ms during the specified period, given by the formula:

*((Number of transactions under 750ms.)/(Total Number of transactions))*100*

Expressing the guarantee as a success rate provides the ability to retain a strict guarantee (target 100%), whilst it also allows for the actual value representing how good or bad the service was.

Using this method, the target is now not an upper limit of 750 ms, but is the ratio to be maintained. In cases where the guarantee must be strict, then the target should be 100% which leaves no place for even a single failure. Note that in this case, an additional variable has been introduced, the Metric Parameter. This parameter should be implemented as a Metric parameter to enable simple modifications if required.

An alternative model to this method may be the forcing of an escalation type model:

The following solutions defines three Metrics instead of a single Metric, as in the previous solutions.

Metric	Successful transaction response time
Target	95
Tracking period	1 month
Unit of measurement	% of success
Metric parameter	750 ms
Metric	Successful transaction response time
Target	99
Tracking period	1 month
Unit of measurement	% of success
Metric parameter	850 ms
Metric	Successful transaction response time
Target	100

Tracking period	1 month
Unit of measurement	% of success
Metric parameter	1000 ms

In a case where it is necessary to report the contractual obligation, as well as the number of transactions that exceed the threshold of 750 ms, you need to define an additional Metric to count the number of failed transactions.

Note: Each Metric generates a single result over a given time period. If it is set to calculate the percentage of transactions, it cannot provide the number of transactions as well.

The only way to produce additional reports from a Metric is to use the outputs from the Business Logic. (Refer to Outputs - User Tables (see page 152) that discusses outputting results from the Business Logic).

Metric	Number of Failed Transactions
Target	No target
Tracking period	1 month
Unit of measurement	Number of transactions
Metric Parameter	750ms
Service	CNP System
Service Domain	Performance
Domain Category	Number of transactions
Timeslot	Always

Case Study 4: Helpdesk Performance

Case Study Illustrating A Helpdesk Situation

The help desk must achieve 100% success in attaining all of the following:

Ticket Type	Resolution Time
Priority 1	1 hour
Priority 2	2 hours
Priority 3	4 hours

Suggested modeling, solution A:

Metric	Resolution time for priority 1
Target	100
Tracking period	1 month
Unit of measurement	% of success
Contract parameter	Resolution time matrix
Service	Helpdesk
Service Domain	Helpdesk performance
Domain Category	Ticket resolution time
Timeslot	Always

The above matrix applies to three Metrics. For each priority, a separate Metric is defined with all of the priorities within the same categories.

Suggested modeling, solution B:

The Metric definition remains the same as shown in solution A.

Option 1:

Service	Helpdesk
Service Domain	Ticket management priority 3
Domain Category	Ticket resolution time
Domain Category	Ticket response time
Timeslot	Always

Option 2:

Service	Helpdesk
Service Domain	Ticket management
Domain Category	Priority 3 ticket resolution time
Domain Category	Ticket response time
Timeslot	Always

Case Study 5: System Backup

Backup is performed as follows:

Timeframe	Number of Backups
weekly	6
monthly	27
yearly	350

Suggested solutions:

Metric	Weekly Backup Count
Target	6
Tracking period	1 week
Unit of measurement	Backups
Service	Backup
Service Domain	Backup
Domain category	Number of backups per week
Timeslot	Always
Metric	**Monthly Backup Count**
Target	27
Tracking period	1 month
Unit of measurement	Backups
Service	Backup
Service Domain	Backup
Domain category	Number of backups per week
Timeslot	Always
Metric	**Yearly Backup Count**
Target	350
Tracking period	1 year
Unit of measurement	Backups
Service	Backup
Service Domain	Backup
Domain category	Number of backups per week

Timeslot	Always

Financial Metric Modeling Example

The following case study presents an example of financial modeling.

Case Study 6: Modeling Financial Conditions of a Contract/Service

There are thee general types of Metrics used for modeling the financial conditions of a service or contract. These are:

- Fixed Price costs
- Consumption costs
- Penalty/Incentive charges

Consider the following example:

A new company is starting out and requires an Email service to be provided along with the setting up and maintenance of its mailboxes. As new staff are employed, the number of mailboxes will obviously increase. The provision of an Email Service for a Contract incurs a fixed price cost of $1000, plus there is an additional cost per mailbox that is to be charged per month. This cost per mailbox is a tiered pricing model as follows:

Number of Mailboxes	**Cost per Mailbox**
1-1,000	$1.00
1,001 - 5,000	$0.80
5,001+	$0.50

So, the more mailboxes that are added, the lower the additional cost. (For example: 1500 Mailboxes will cost (1000 x $1) + (500 x $0.80) = $1400.) Using this model, two Metrics can be produced to reflect this in the contract.

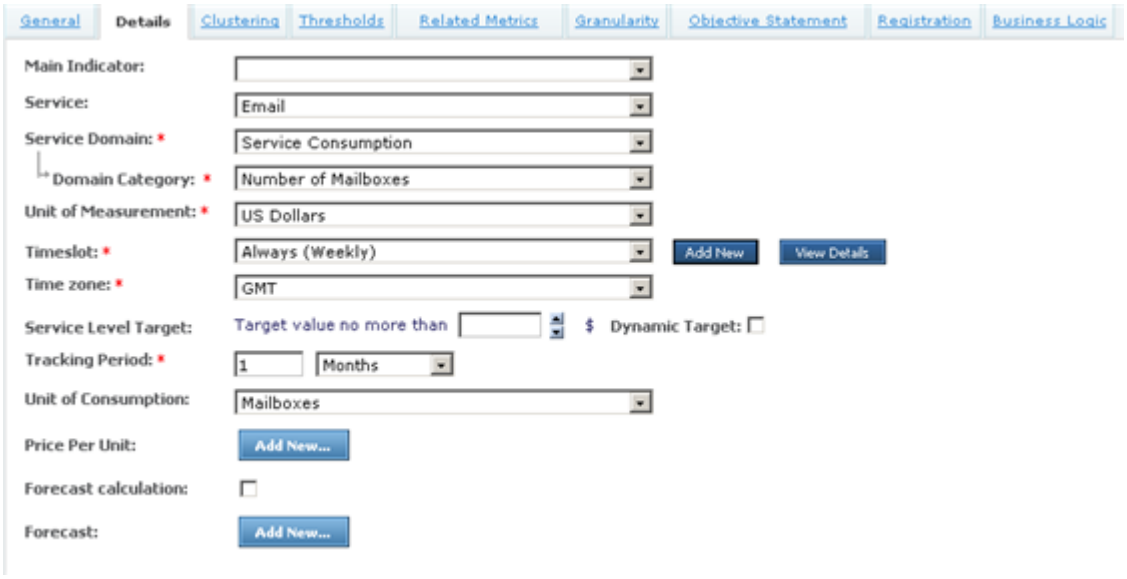
- Email Service Cost (Fixed)
- Mailbox Usage Cost (Variable/Consumption based)

In addition there is an estimate by the management team of the number of staff members throughout the year (2007) as follows. The trend is due to initial company growth through employment and then from new offices opening in other regions:

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sept	Oct	Nov	Dec
50	100	500	900	1600	1700	1800	2500	2600	3500	3600	5800

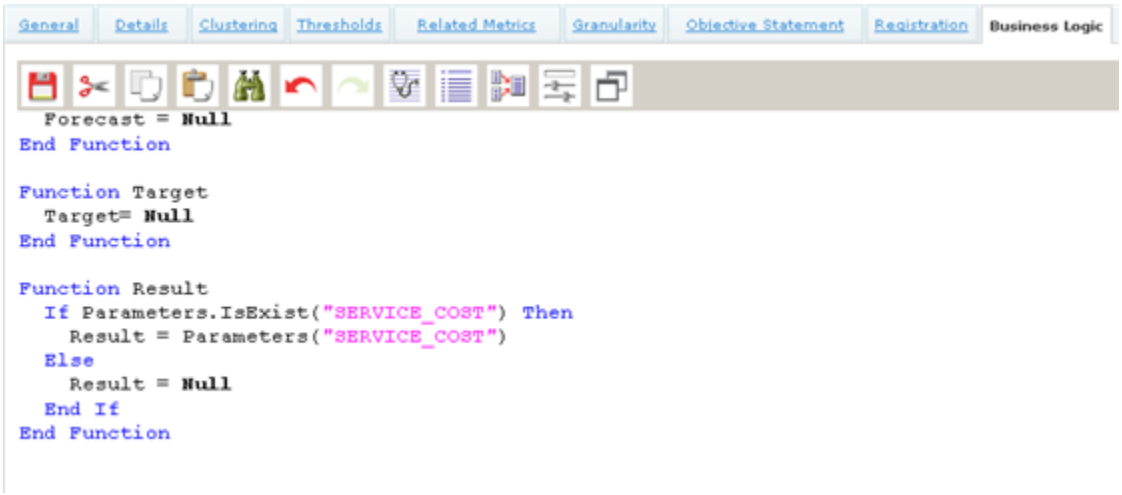
To model these Metrics, do as follows:

Create the Fixed Cost Metric (use the Price Item type) in the contract, using the following details:

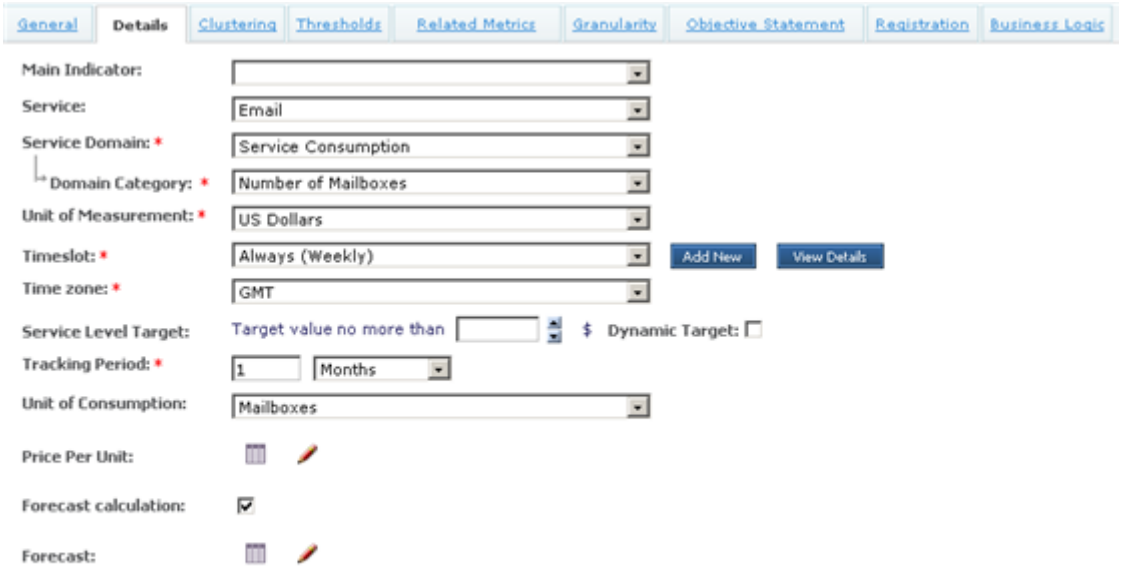


In order to specify the fixed cost to the Contract of this service, implement this as a Parameter to the Business Logic (where the fixed cost must be returned from the Result function) This parameter can then be exposed via the Objective Statement of the Metric, as shown below:

Returning the parameter value for this Metric, is simply a case of returning the value of the service cost via the Result function..

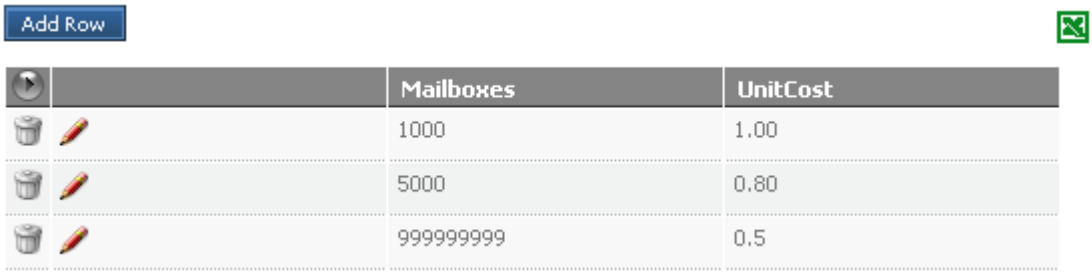


Next, create the variable pricing Metric (again, use Price Item Type) to determine the consumption costs of the number of mailboxes used. Name this Metric '*Mailbox Consumption Cost*' and create it with the following details:



In this instance, you need to enter the consumption parameters into the Metric details. These will go into the Price Per Unit table. To model the table above for the number of mailboxes against the cost, create a column for the upper limit of mailboxes and one of the unit prices:

Then enter the values for each tier. In this case, the upper limit of mailboxes determines the cost bracket associated with it. Since there are 3 tiers, they are added to the table in this manner:



In addition to this, implement the forecasting function on the consumption of Mailboxes. Do this by creating the Forecast table with the preset Monthly layout.

This is then filled with the values from the tables given in the scenario description.

Consumption	Jan-01-2007	Feb-01-2007	Mar-01-2007	Apr-01-2007	May-01-2007	Jun-01-2007	Jul-01-2007	Aug-01-2007	Sep-01-2007	Oct-01-2007	Nov-01-2007	Dec-01-2007
1	50	100	500	900	1600	1700	1800	2500	2600	3500	3600	5800

Now you can add the Objective statement for the Metric. In this case, no parameter values are required since they are derived from the 'Price Per Unit' and 'Forecast' tables.

Finally, complete the Business Logic as follows:

```
Option Explicit

Dim PPUmap1, PPUmap2, PPUmap3, PPUkey, FCmap, periodFC, TierPPU
Dim currentMonth, TotalMailboxes, MailboxesThisPeriod, TotalPrice

Sub OnRegistration(dispatcher)
   'sample registration only
   dispatcher.RegisterByMetric "OnMailboxAddedEvent", "NewMailboxEventType", _
      "MailboxResource", "MONTH", "MetricName", "MetricContract", _
      "MetricContractParty"
End Sub

Sub OnLoad(TIME)
   'Initialise the price tier maps and forecast maps
   Set PPUmap1 = Context.Field ("Price Per Unit")(1)
   Set PPUmap2 = Context.Field ("Price Per Unit")(2)
   Set PPUmap3 = Context.Field ("Price Per Unit")(3)
   Set FCmap = Context.Field ("Forecast")(1)
End Sub

Sub OnPeriodStart(TIME)
   'TODO: ADD code here TO handle period START event
   currentMonth = GetMonth (time)
   If Context.IsInForecast Then
      periodFC = getForecastValue (currentMonth)
   End If
   MailboxesThisPeriod = 0
   TotalPrice = 0
End Sub

Sub OnPeriodEnd(TIME, isComplete)
   ' Calculate the current price of all the mailboxes using the tiered
   ' pricing model
   ' This uses a cumulative approach as it goes through each tier to
   ' determine total cost.
   TotalPrice = getMailboxCost (TotalMailboxes)
End Sub

Sub OnTimeslotEnter(TIME)
End Sub

Sub OnTimeslotExit(TIME)
End Sub

Sub OnMailboxAddedEvent(eventDetails)
   MailboxesThisPeriod = MailboxesThisPeriod + 1
   TotalMailboxes = TotalMailBoxes + 1
End Sub
```

```
Function Forecast
   Forecast = getMailboxCost (periodFC)
End Function

Function Target
   Target = Null
End Function

Function Result
   result = TotalPrice
End Function

Function getforecastvalue(q)
   getforecastvalue = FCmap (q)
End Function

Function getmonth(time)
   'this function retrieves the month
   Dim lTime
   lTime = Tools.GetLocaleTime(time)
   getmonth = monthname (datepart ("m", lTime), True) & _
      "-0" & datepart ("d", lTime) & "-" & datepart ("yyyy", lTime)
End Function

Function getMailboxCost(num_boxes)
   'Function calculates the cost of the mailboxes using the tiered pricing model
   Dim returnValue
   If num_boxes <= PPUmap1 ("Mailboxes") Then
      'First tier
      returnValue = num_boxes * PPUmap1 ("UnitCost")
      'Out.Log "Tier1: " & num_boxes
   Else If num_boxes > PPUmap1 ("Mailboxes") And num_boxes <= PPUmap2 ("Mailboxes")
Then
      'second tier only
      returnValue = (PPUmap1 ("Mailboxes") * PPUmap1 ("UnitCost")) + _
         ((num_boxes - PPUmap1 ("Mailboxes")) * PPUmap2 ("UnitCost"))
      'Out.Log "Tier2: " & num_boxes
   Else If num_boxes > PPUmap2 ("Mailboxes") Then
      'third tier
      returnValue = (PPUmap1 ("Mailboxes") * PPUmap1 ("UnitCost")) + _
         ((PPUmap2 ("Mailboxes") - PPUmap1 ("Mailboxes")) * PPUmap2 ("UnitCost")) +
_
         ((num_boxes - PPUmap2 ("Mailboxes")) * PPUmap3 ("UnitCost"))
      'Out.Log "Tier3: " & num_boxes
   End If
   getMailboxCost = returnValue
   'Out.Log "Cost is: " & returnValue
End Function
```

Note: This Business Logic script handles both the forecast calculation (using the 'Forecast' table) and the Financial Consumption Cost results. Both use of the same formula getMailboxCost() which calculates the tiered pricing based upon the 'Price per Unit' table defined for this Metric.

Data Modeling Examples

The following two cases illustrate the basic Data Modeling process, as well as some of the finer points that may be involved.

Because the process of Data Modeling is done after the process of Contract Modeling, the calculation requirements derived from the contract's guarantees are already known, having been identified and defined as part of the Contract Modeling process.

The data model must take into account all calculation requirements.

In the following two case studies, a single or select few requirements are chosen in order to demonstrate the Data Modeling process.

Case Study 7: Server Performance

This is a typical case study for server performance.

Given the following data source structure:

Indication	Server	Measure	Timestamp
availability	Appserv01	1	03/01/2001 07:44
response time	Appserv01	354.6	03/01/2001 09:58
CPU load	Dbserv02	83%	03/01/2001 12:12
availability	Appserv01	0	03/01/2001 14:26
CPU load	Dbserv02	94.30%	03/01/2001 16:40
capacity	Firewall01	10%	03/01/2001 18:54
response time	Dbserv02	476.89	03/01/2001 21:08
availability	Appserv02	1	03/01/2001 21:24
response time	Appserv01	774.4	03/01/2001 21:48
CPU load	Dbserv01	83%	03/01/2001 21:52

In addition to the above, are the following calculation requirements:

Calculate the % availability of each application server.

The availability of each server should be calculated separately. Therefore, to calculate availability for a single server, it is necessary to receive events for only this specific server. In addition, the data sources contain other performance indicators that are not relevant to availability calculations (response time, capacity, and so on), so the availability indicators as well as the relevant server need to be filtered.

Since the filtering criteria in CA Business Service Insight are Event Type and Resources, translate the filtering criteria from the data source values into a definition of Resource and Event Type.

In this case, the indicator is an ideal value to translate as an Event type in CA Business Service Insight since it describes logically the type of Event. There are a limited number of types, such as, availability, response time, capacity, and CPU load. This means that for the Metrics that calculate availability for the server, the registration is for the availability Event Type.

In this case, when there are a large number of servers and it is necessary to calculate the availability of each server, it follows that each server has to be defined as a Resource. It then needs to be grouped within a Resource group and the Metric will be clustered over that Resource group.

Suggested modeling:

Event name	Availability Event.
Behavior	Reported as change of status of 0 or 1.
Timestamp field	Timestamp (the only time field in the data source).
Resource field	Server (every server that appears in the data source is translated into an CA Business Service Insight resource).
Event type field	Indicator (each of the values in this field are translated into an Event type in CA Business Service Insight. There are four Event types).
Data fields	Measure is 0 or 1 (only for availability records).

The following resource allocations should be defined:

Resource Type attribute	Application Server
Allocation to Contract party	Each application server is allocated to the contract party, where the relevant server runs its application. This enables registration by contract party that retrieves all servers accordingly.
Allocation to Service	Same as above.

Allocation to Resource group	Optional. It is usually necessary to group Resources where clustering is required.

And lastly, based upon all of the above definitions;

Registration by	For the clustered Metric, calculating the availability of each server individually, the registration is by Resource.

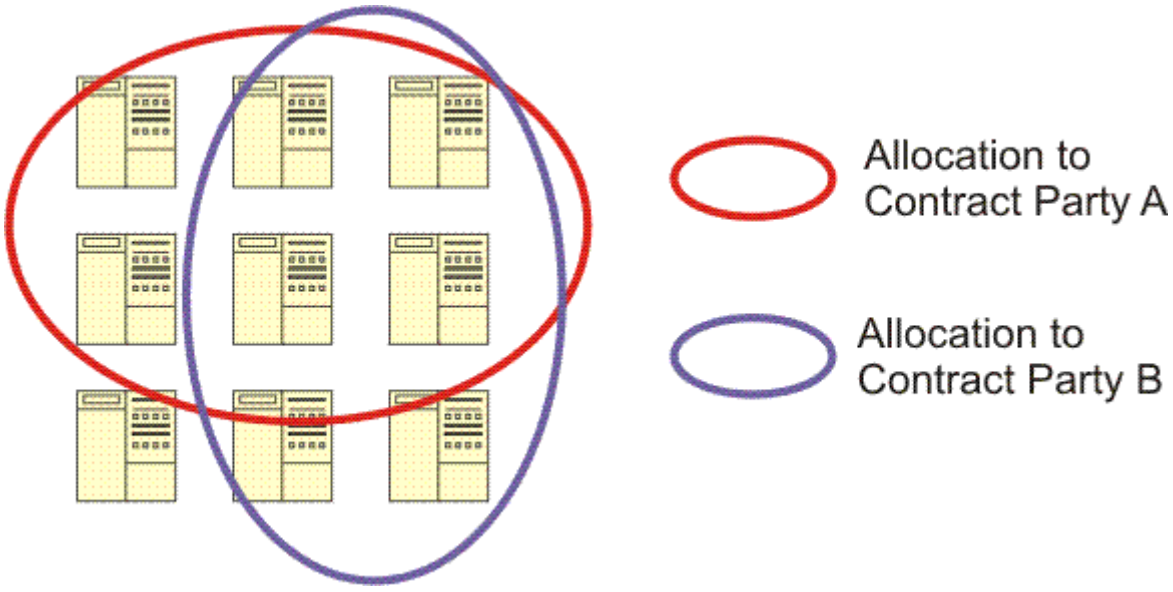
In order to be able to meet the above requirement the following criterion is added:

Calculate average response time of application servers for each Contract party separately.

For this requirement, it is necessary to receive Events of response times for all application servers that are part of the group of servers running applications for the specific contract party. Receiving the response time events is done by registering the Event Type created from the indication field with the value "response time". This ensures receiving only events that report on response times of servers.

In order to receive only the Events that report on the relevant servers for a specific contract party, the resources need to be registered through the Contract party allocation.

A resource can be allocated to more than a single contract party, Service or group. Therefore, it may happen that an event that was sent for the calculation of response time as part of the Contract of contract party A, is also a part of the calculations for contract party B.



Case Study 8: Helpdesk Performance

This is a typical case study for Helpdesk performance

Given the data source shown below:

TK No.	TK. Priority	Raised at	Closed on	Resolved at	Cust Ref	Location
3800968	1	19/12/2003 07:51	05/01/2004 11:31	22/12/2003 12:00	CM3	London
38000509	1	18/12/2003 09:21	05/01/2004 11:33	22/12/2003 12:00	CM1	Ipswitch
38084199	2	07/01/2004 11:20	14/01/2004 09:10	09/01/2004 12:00	CM2	Ipswitch
38188329	3	21/01/2004 09:27	27/01/2004 09:09	24/01/2004 12:00	CM3	Leeds
37964069	3	12/12/2003 14:04	05/01/2004 11:35	19/12/2003 12:00	CM286	Birmingham
3796448	1	12/12/2003 14:18	05/01/2004 11:39	19/12/2003 12:00	CM263	Luton
37965039	2	12/12/2003 14:57	14/01/2004 15:05	18/12/2003 12:00	CM264	Leeds
37970699	2	15/12/2003 09:26	05/01/2004 11:22	22/12/2003 12:00	CM288	London
37997259	1	17/12/2003 15:58	05/01/2004 11:27	23/12/2003 12:00	CM302	Ipswitch
38000259	1	18/12/2003 09:11	06/01/2004 14:44	29/12/2003 12:00	CM340	London
38021049	1	22/12/2003 09:32	06/01/2004 14:28	31/12/2003 12:00	CM341	London

The data source shown above lists details for helpdesk tickets that are managed for each customer in the various locations that the customer services. A single location may also be shared between customers.

The following three requirements are required for reporting when using this data source:

- % priority 1 tickets resolved within four hours for customer CM3.
- % priority 1 tickets resolved within four hours for customer CM3 at each location.
- % priority 1 tickets closed within one day for customer CM3 at each location.

The above requirements indicate that the Events should be filtered by:

- priority
- customer
- location

You need to specify which of these filtering criteria is translated as an Event Type and which is to be the relevant Resource.

How do I select an Event Type?

Of the three filtering criteria needed, the ticket priority is the most appropriate to be translated into the Event type for the following reasons:

- It describes the kind of event that is being handled (Priority 1 tickets).
- The number of different priorities that exist is relatively small (priority 1,2,3).
- The event type itself is relatively constant (the helpdesk seldom changes the priorities by which it handles tickets).

How is a Resource selected?

The two other filtering criteria required, are the Customer and Location and these are the smallest entities that require reporting. Therefore, the combination of customer and location is the Resource.

Customer and location are relatively fixed entities and have a definite life-cycle, whereby new customers or new sites may be added. Additionally, the relationship between a site and a customer may change.

For the purposes of translations, it is possible to use more then one field from the data source. Whereas in the previous case study the server field was translated into an CA Business Service Insight Resource, in this case the Resource is built using the combination of two fields. Each permutation therefore produces a new Resource.

The Resources list is shown below:

Customer Field	Site field	Output Resource
CM3	London	CM3@London
CM1	Ipswitch	CM1@Ipswitch
CM2	Ipswitch	CM2@Ipswitch
CM3	Leeds	CM3@Leeds
CM286	Birmingham	CM286@Birmigham
CM263	Luton	CM263@Luton
CM264	Leeds	CM264@Leeds
CM288	London	CM288@London
CM302	Ipswitch	CM302@Ipswitch
CM340	London	CM340@London
CM341	London	CM341@London

The output Resource is a combination of the customer and site fields using the symbol '+' to combine them. It is important to be aware of the name of the Resource since it is extracted from the data source and it appears later in the reports. The reporting abilities need to meet expectations.

Note: One of the most common mistakes when modeling a helpdesk or any ticket or incident system is to define a single incident as a Resource.

The following incorrect assumptions often lead to mistakes:

1. The single incident is that which is reported.Do not set the entity to be reported as the entity that calculations are generated for so that the single incident is not the basis of the calculations for the customer site. In general, the SLA is based upon overall performance, not the individual ticket handling performance.
2. The guarantee is by ticket level. This is a mistake because the guarantees are periodic ones, what is calculated is an aggregation over time.

Setting Allocations for the Resources

For the first calculation requirement:

1. % Priority 1 tickets resolved within four hours for customer CM3.
   - Events of tickets attributable to a specific customer need to be received. The customer is part of this Resource, which also indicates the customer site. Therefore, to capture all Events related to Resources and attributable to that customer, there are two options:
   - In cases where the customer in the data source represents a Contract party, the Resources can be attached to the relevant Contract party. This allows registration by Contract party. Wherever possible, this is always preferable.
   - Create a Resource group for each of the customers in the data source and group all relevant Resources within it. Using this method, the registration is by Resource group.

   For the next two calculation requirements:
2. % Priority 1 tickets resolved within four hours for customer CM3 at each location.
3. % Priority 1 tickets closed within one day for customer CM3 at each location.

For these requirements, gather the Resources into Resource groups because the Metrics have to be clustered given that the calculation is required for each site individually.

Note: Even if the allocations of Resources for the current model and requirements are not necessary, it is important to create them keeping in mind future requirements. Doing so prevents overhead in future system development.

Choosing the Timestamp field

As mentioned previously, the Timestamp is very important to the way in which the correlation Engine handles the Events. Metrics always calculate service level per time period, and the Events that are processed within this time period are the ones whose Timestamp falls within the period.

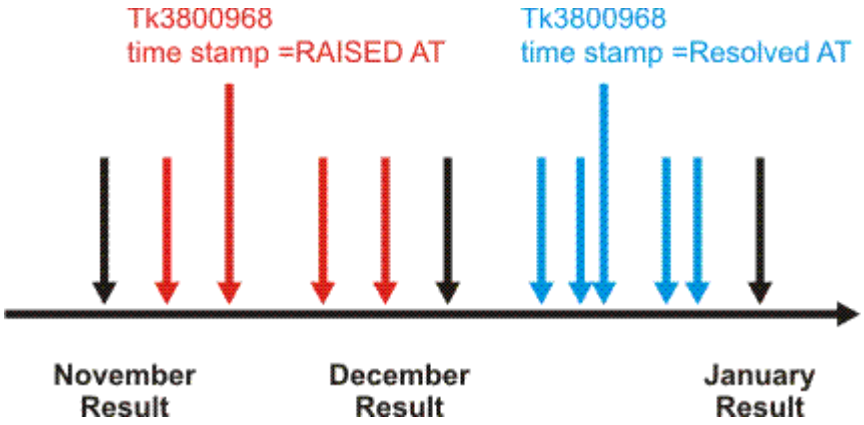
In the first case study, the data source only has one time field. However in this case there are three possible fields that that can be set as the timestamp. Examining the first two records:

TK No.	TK. Priority	Raised at	Closed on	Resolved at	Cust Ref	Location
3800968	1	19/12/2003 07:51	05/01/2004 11:31	22/12/2003 12:00	CM3	London
38000509	1	18/12/2003 09:21	05/01/2004 11:33	22/12/2003 12:00	CM1	Ipswitch

To calculate the resolution time, both the Raised at time and the Resolved at time are required. For the purposes of the Event, it is possible to attach only one Timestamp to it. Then the other can be used as a value within the value fields.

If the Raised at value is used as the Timestamp, then the ticket is included in the December results. If the Resolved at value is used as the Timestamp, then the ticket is included in the January results. Both options are viable. The selection just needs to meet expectations as to where the tickets should appear in reports.

It is very important point to consider during implementation, since it determines when events are used for calculations. For example, if a ticket is raised in November, but is not closed until December, when should it contribute towards the SLA result? Does it go into the November data, or December?



In this case, since the ticket is reported to the data source only after it is closed, the data can be captured only after the ticket is closed. Usually the Closed date is after both the Raised and Resolved dates. In a case where the Raised date was chosen to be the Timestamp, then it should be processed as part of the December results. The Closed date was in January, which means that December had already passed when this ticket was reported. Hence, the results for December had already been published. The correlation engine then notices the Event as being in the past because the Timestamp belongs to December and triggers recalculation. Therefore, the results of December change retroactively.

These consequences need to be understood completely in order to be able to define which time field must be chosen as a Timestamp. Typically, the Closed date is chosen in order to have final reports that do not change retroactively.

On the other hand, using the Closed date as the Timestamp delays tickets from entering calculations. A ticket that has been resolved may only be closed at a substantial time later.

Be aware however, that this use of the Closed date might also trigger a process in the organization that accelerates the closing time of tickets.

The final suggestion is then:

Event name	Priority 1 ticket (can also be defined for other priorities if required)
Behavior	Reported when ticket is closed
Timestamp field	Closed at
Resource field	customer field+site field
Event type field	Priority
Data fields	All
Resource Type attribute	Contract Party Site
Allocation to Contract party	Each site is allocated to relevant Contract party
Allocation to Service	Same as above
Allocation to Resource group	Resource group is created for each Contract party to enable clustering
Registration by	For clustered Metrics, registration is by Resource and Metric is attached to a Resource group called Customer CM3 sites For closed time Metrics, registration is by Contract party and Service

Using Custom Attributes Example

The following case study presents an example for dynamic multiple targets.

Case study 9: Dynamic Multiple Targets

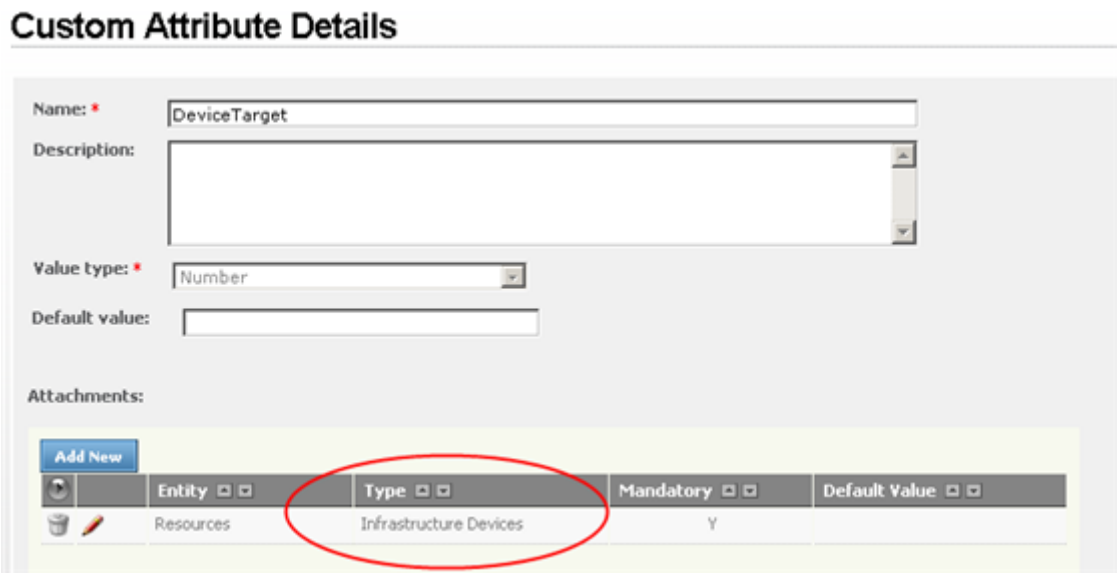
Consider an example scenario where all of the hardware infrastructure devices in a customers' environment have individual targets set upon them for their availability requirements. Using the standard modeling approach this would be quite a difficult task to achieve and would involve a lot of logical grouping for the devices and management using the resource model. To add to the complexity, the targets for these devices can change over time. These target values are updated in CA Business Service Insight by a translation script as the details are stored in an external CMDB (see Translation Script Best Practice Examples (see page 245) for the translation script example)

In this instance the Metric could be as follows:

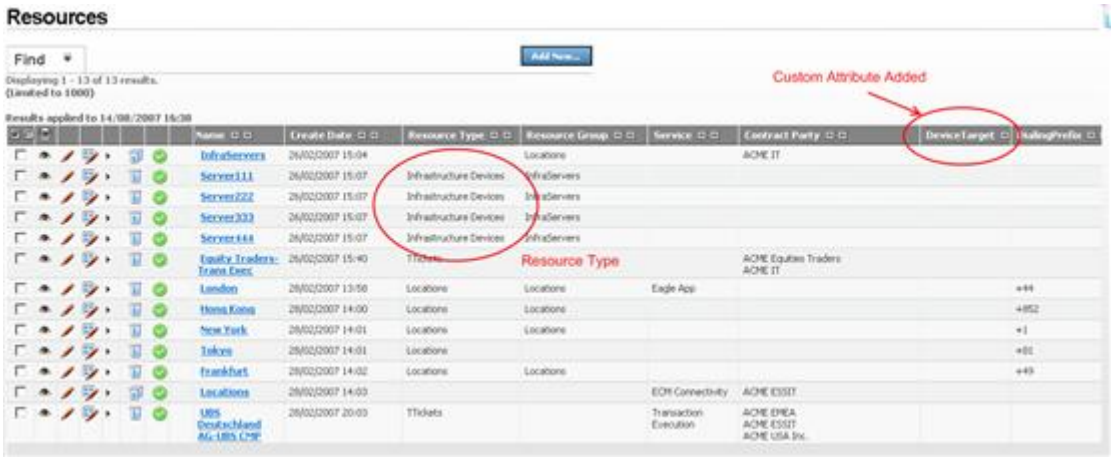
% of availability per hardware device.

One way to effectively model this is to use the 'Custom Attributes' feature along with one of the other key features, 'Dynamic Targets'. Both of these can be used with a clustered Metric to achieve the desired results. Adding the service level target to the resource directly allows the business logic to compare the service level of each resource (hardware device) against its own target. A clustered Metric provides the individual service compliance for each piece of hardware using a single Metric.

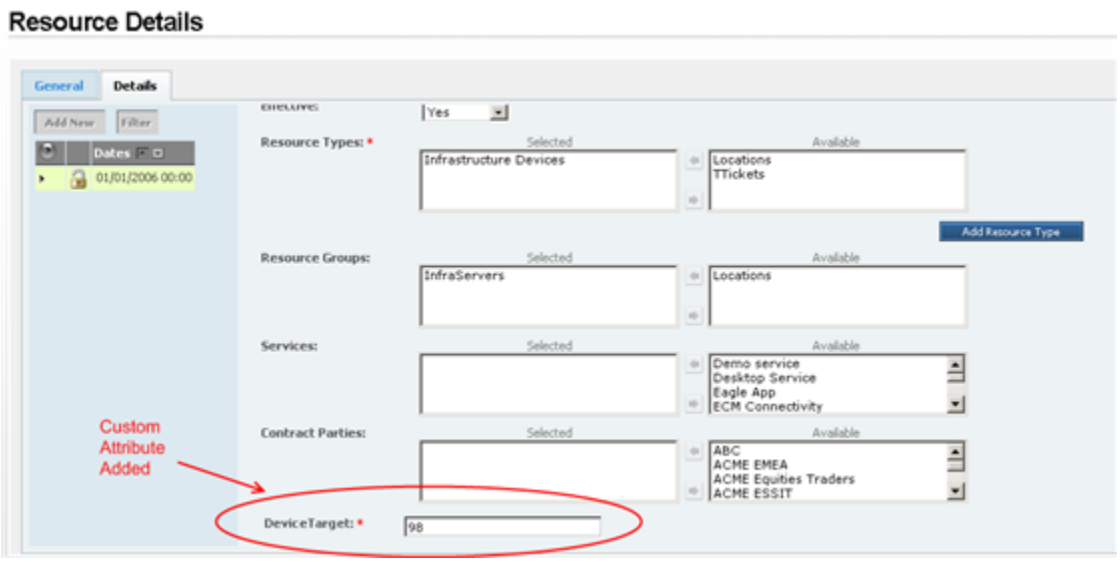
Therefore, it is necessary to first create the custom attribute by adding it to the resource type of these devices (where all devices are a resource of type 'Infrastructure Device'). The custom attribute created is called the 'DeviceTarget', and can be added from the menu at *Service Catalog > Custom Attributes*. Note, that when creating the custom attribute, you must link it to the resource type(s) which require it.



Now, when viewing the resources in the system, it can be seen that the new custom attribute is available for the resource type that it was linked to.



And the individual resources have a new field which can be updated.



In this example, this field would normally be inserted/updated by the translation script.

Now that each of the resources has a target specified against it, you can develop the logic to perform the required calculation (after committing the resource changes). The following sample code shows how to extract the custom attribute value from the resource (in bold).

```
Option Explicit

Dim TotalTime
Dim OutageTime
Dim PeriodStart

Sub OnRegistration(dispatcher)
   dispatcher.RegisterByResource "OnDeviceOutageEvent", "DeviceOutageEvent", _
      Context.ClusterItem
End Sub

Sub OnLoad(TIME)
   TotalTime = 0
   OutageTime = 0
End Sub

Sub OnPeriodStart(TIME)
   TotalTime = 0
   OutageTime = 0
   PeriodStart = TIME
End Sub

Sub OnPeriodEnd(TIME, isComplete)
   TotalTime = Tools.NetTime(PeriodStart, TIME)
End Sub

Sub OnDeviceOutageEvent(eventDetails)
   OutageTime = OutageTime + Tools.NetTime (eventDetails ("OutageStartTime"), _
      eventDetails ("OutageEndTime"))
End Sub

Function Target
   Target = eventDetails.CustomAttribute ("DeviceTarget")
End Function

Function Result
   If TotalTime > 0 Then
      Result = (TotalTime - OutageTime) / TotalTime
   Else
      Result = Null
   End If
End Function
```

Translation Script Examples

The following case studies include example of basic automatic translation and resource model updates.

Case Study 10: Basic Automatic Translation

The Translation script sample provided here is a fairly simplistic case of processing the Pending entries in the Translation Entries screen.

The OnTranslationEvent handler performs a simple check on the first character in the resource, and performs an action according to the value: If 'a' the translation entry is set to 'ignore', if 'b' it is deleted, 'c' it will be translated, or otherwise it is left unchanged to be manually translated. Note that the counters throughout the code keep track of what actions are performed during the script execution. This is very useful for debugging or documentation of the script executions each time it is run, especially when the script is automated. The *Tools.Commit* command at the end of the function is very important to remember, since without it none of the changes made by the script will be saved in the database.

The TranslateResource() function called, simply checks to see if a resource by the same name as the one passed to it by the pending translation entry (along with the E2E- prefix) exists in the system. If it does not, the script adds this resource and then performs the translation. If it already exists, then it creates a translation entry from the resource string to the existing CA Business Service Insight resource.

The final Result function of the script simply outputs a description of the tasks performed by the script. The code is as follows:

```
Option Explicit

dim translated
dim ignored
dim deleted
dim manually
dim ActionDate

Sub OnLoad()
   'tools.log "Translation Script: In OnLoad procedure", "I"
End Sub

Sub OnTranslationEvent(entryDetails)
   Dim dump
   dump = entryDetails.Dump
   tools.log dump

   Dim resource, entryId
   entryId = entryDetails.EntryId
   resource = entryDetails.FieldValue(1)
   ActionDate = entryDetails.LastActionDate

   If mid(resource,1,1) = "a" Then
      tools.IgnoreEntry entryId
      ignored = ignored + 1
      tools.log "ignored" & entryId & " " & resource
   Else If mid(resource,1,1) = "b" Then
      tools.DeleteEntry entryId
      deleted = deleted + 1
      tools.log "deleted" & entryId & " " & resource
   Else If mid(resource,1,1) = "c" Then
      TranslateResource resource, entryId
      tools.log "translated" & entryId & " " & resource
   Else
      tools.SetManualTranslationEntry entryId, 1
      manually = manually + 1
      tools.log "manually" & entryId & " " & resource
   End if

   Tools.commit
End Sub

Sub TranslateResource(resource, entryId)
   Dim newName
   Dim vector
   newName = "E2E-" & resource
```

```
   if NOT tools.IsResourceExists(newName) Then
      Dim resourceDetails
      set resourceDetails = tools.GetResourceDetailsByDate(resource,ActionDate)
      resourceDetails("ResourceName") = newName
      resourceDetails("ResourceTypes") = "E2E Transactions"
      tools.AddResourceByMap resourceDetails
   end if

   tools.TranslateEntry entryId, newName
End Sub


Sub Main()
end Sub


Function Result
   Result = translated & "entries were translated, "& _
      ignored & "entries were ignored," & _
      deleted & "entries were deleted and "& _
      manually & "entries were set to manually update!"
End Function
```

Case Study 11: Resource Model Updates

Another use of translation scripts is to update the CA Business Service Insight resource model with values taken from an external data source. While strictly not related to translation any longer, this sample is a highly valuable facility for automatic updates to the system.

The previous section on Custom Attributes described a scenario where the hardware infrastructure devices of an organization are stored in an external CMDB, along with an expected availability target for each device. This information needs to be replicated into the CA Business Service Insight resource model so as to keep the infrastructure mapping (and the device targets) up to date.

In this case, the script is required to perform the following tasks:

- Add any new hardware devices which do not currently exist in the system.
- Update the expected availability target of devices already existing (if the target is changed).

The script is as follows:

```
Option Explicit

'*********************************************************************
'Global Variables and constants
'*********************************************************************

Dim added
Dim updated
Dim ChangeSetName

added = 0
updated = 0

Const RESOURCE_TYPE = "Infrastructure Devices"
Const RESOURCE_GROUP = "InfraServers"
Const CHANGESET_NAME = "Infrastructure Devices"
Const CHANGESET_EFFECTIVE_DATE = "01/01/2007"

'*********************************************************************
'Sub OnLoad :
'Preparing the foundation infrustructure enteties
'*********************************************************************

Sub OnLoad()
   Tools.log "Translation Script : In OnLoad procedure", "D"

   'Search for existing preliminary resource version
   Dim ChangeSetMap
```

```
   Set ChangeSetMap=Tools.SearchChangeSets(CHANGESET_EFFECTIVE_DATE,
CHANGESET_NAME)

   'If no existing version create a new version
   If ChangeSetMap.EMPTY Then
      Tools.AddChangeSet CHANGESET_NAME, CHANGESET_EFFECTIVE_DATE, CHANGESET_NAME
      Tools.Log "Changes set '" & CHANGESET_NAME & "' added."
   End If

   Set ChangeSetMap = Nothing
End Sub

Sub OnTranslationEvent(EntryDetails)
End Sub

Sub Main()
   Dim conn, rs, resource, deviceTarget, resource_id, resMap, custAttrib,
custAttribValue
   Set conn = CreateObject("ADODB.Connection")
   Set rs = CreateObject("ADODB.RecordSet")
   conn.open "DSN=HardwareDevices"
   rs.Open "select * from tblServers", conn

   Do While Not rs.EOF
      resource = rs("serverName")
      deviceTarget= rs("DeviceTarget")
      'Add resources to latest version if it doesnt exist already
      If Not Tools.IsResourceExists(resource) Then
         resource_id = Tools.AddResource(resource, CHANGESET_NAME, "Infrastructure
Device", RESOURCE_TYPE, RESOURCE_GROUP, Null, Null)
         Tools.UpdateResourcesCustomAttribute resource, CHANGESET_NAME,
"DeviceTarget", deviceTarget
         Tools.Log "AddingResource: " & resource & " with target: " & deviceTarget
& " ; assigned ID= " & resource_id
         added = added + 1
      Else
         Set resMap = Tools.GetResourceDetails(resource,CHANGESET_NAME, False)
         Set custAttrib = resMap("CustomAttributes")
         custAttribValue =
CDbl(custAttrib("DeviceTarget")("CustomAttributeValue"))
         If CDbl(deviceTarget) <> custAttribValue Then
            Tools.UpdateResourcesCustomAttribute resource, CHANGESET_NAME,
"DeviceTarget", deviceTarget
            Tools.Log "Updating Resource target for : " & resource & " from: " &
deviceTarget & " to " & custAttribValue
            updated = updated + 1
         End If
      End If
```

```
      Tools.Commit
      rs.MoveNext
   Loop

   rs.Close
   conn.Close
   Set rs = Nothing
   Set conn = Nothing
End Sub

Function Result
   ' Commit the transaction
   Tools.CommitChangeSets CHANGESET_NAME
   Result = added & " resources added and " & updated & " resources updated."
End Function

'*******************************************************************************
```

Business Logic Scripting Examples

Here are a number of general guidelines for Business Logic Scripting:

Global variables

- Be sure to initial the global values you declare. The PSL state mechanism cannot save variables with null values

General coding

- Before using one of the objects listed below, verify that it exists (using a suitable IsExist method):
  - All types of parameter (e.g. table, list, etc.)
  - Custom attribute
  - Resource
- Make sure you provide a Business Logic Module with all the parameters it requires
- Before changing a resource name verify which Metrics are using it and update them accordingly

Maps

- Use of maps in clustered Metrics: Maps require more calculation effort from the engine, be aware that when you have clustering on a Metric you multiply the effort by the number of cluster items.
- Large global maps within the Business Logic of clustered Metrics should be used only after careful consideration. While the Engine is calculating a clustered Metric it is busy loading the global variables from the states for each item in the cluster separately
- Be sure to clear out maps and vectors after you are finished with them
- If required to use large maps, make sure you manage the map efficiently by dividing it into logical ranges.

Case Study 12: Using Counter Logic to Calculate Number of Failures

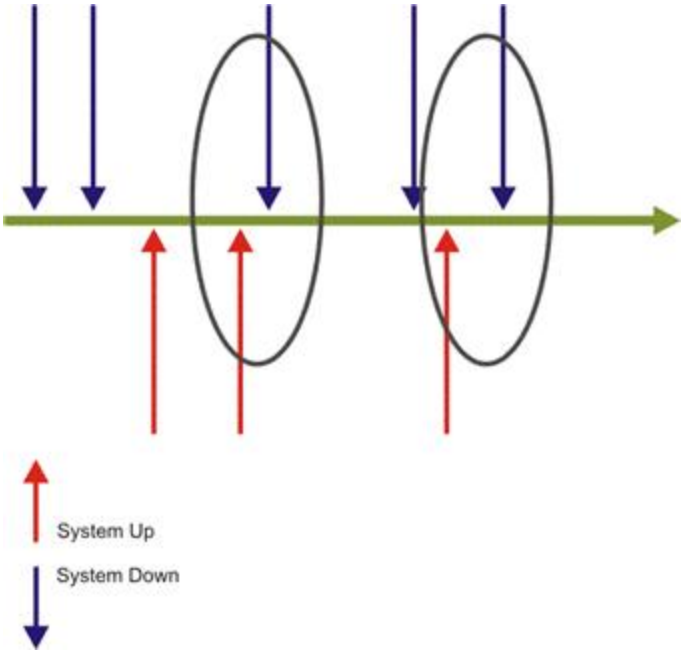
The following example calculates the number of failures that occurred within a given calculation period. This formula and the methods used to implement it can be taken as an example of a formula that is required wherever it is necessary to calculate something.

The following calculation assumptions are used:

- Input Events:
  - Availability Event, Status (0/1)
  - The availability Event arrives every few minutes. It is not possible to assume the frequency of the Events (the Event may occur every five minutes or once an hour) and there may also be redundant Events (for example: 0, 0, 0, 1, 1, 0, etc.).
- Timeslot- Business Hours, Failures that occurred out of timeslot are not counted.
- Required Result is the count of how many failures have occurred during the period.

In order to tally the failures that occurred during the calculation period it is necessary to store a periodic counter variable and also a variable that stores the system status. Since redundant Event information may be received (i.e. An 'Up' event followed by another 'Up' event), it is also necessary to count the number of locations where there was a change of the system status from "Up" to "Down" and not to count every time a "Down" event is received as this may be a redundant Event that represents a failure that has already been counted.

The following figure graphically depicts counting system Up and Down times.



Important points that should be considered:

- **Constants**-the use of constant definitions, rather than that of constant values within the code, is recommended. In this way, if the value needs changing, it is easy to change the value in the constant definition only and not to have to search for it throughout the code. In addition, it is easy to change it to a parameter use if required. In the above example, the values that represent the system status received in the event are defined as constants in order to make the code more understandable, and also to differentiate when the number zero is used as a number for counting purposes rather than where it represents a system status.
- **Global variables**:
  - **Counter**-the definition of the counter variable is a global definition. In the formula it is declared at the top of the code, and outside of any subroutines or functions. It is important to define the counter variable in this case as a global variable. This allows it to be used in several subroutines/functions within the code, and also allows the system to keep its value throughout the calculation period and provide its result at the end of the period.

    In this example the counter variable needs to be used in three places in the code:

    - To be initialized at the beginning of a period.
    - To be advanced in case of a failed event in the event handler.
    - To be outputted by the result function at the end of the period.
  - **System previous status**-this variable holds the status of the system and is updated whenever a new event with the system status is received. This variable also needs to be global because it is used in several subroutines/functions in the code, such as:
    - To be initialized in the beginning of the calculation.
    - To be updated when a new event is received.
- **Timeslot considerations**-in order to verify whether a failure occurs within a timeslot period, it is possible to use the value of the *context.IsWithinTimeSlot* property. The context is a global object that can be used from anywhere in the code, and in this case, is used to indicate when a failure is received and whether it is within or outside of the timeslot period. If at the timestamp of the event the flag returns TRUE, then the event occurs within a timeslot period, however if it returns FALSE, this indicates that the event took place outside of the timeslot. According to the required logic, any failure that occurs outside of the timeslot is ignored and therefore not counted.
- **Initialize variables**:
  - **Counter variable**-holds a periodic value and is therefore initialized in the *OnPeriodstart* handler to make sure that each calculation period counts its failures separately. Each period starts with zero and outputs only the number of failures within this period.

In cases where it is required to calculate the accumulated failures within each period, (meaning that the result of each period is all of the failures that have occurred up until the end of this period, and including all periods before that), then it is required to initialize the counter only in the *OnLoad* handler and remove it from the *OnPeriodStart* handler. Thus, the counter continues counting and accumulating between the periods as shown below:

```
Sub OnLoad(time)
    FingerprInted=0
End Sub
```

- **System status variable**-should be initialized once at the beginning of the calculation, and from that point on, be updated on each Status event. This variable retains its value throughout the calculation and is not restricted to a certain calculation period. It must also retain its value between the calculation periods. Since the status of the system before receiving the first Event is unknown, an assumption must be made as to the system status. The common assumption is that the system is "Up". This assumption should be agreed upon and checked as it can lead to the following:

  If when the calculation started and the system status was in fact "Down" and the first event arrives to indicate this "Down" status, this will be counted as a failure because of the assumed status was "Up". This failure will be counted towards the first calculation period even if it did not necessarily occur during that period.

```
Option Explicit

'Constants definitions
Const UP=1
Const DOWN=0

'Global variable for counting failures
Dim FingerprInted
Dim SystemStatus

Sub OnRegistration(dispatcher)
   ' The parameters of the method are: <procedure name>, <Event Type>
   dispatcher.RegisterByContractPartyAndService
"OnAvailabilityEvent","AvailabilityEvent"
End Sub

Sub OnLoad(time)
   SystemStatus = UP 'assume the first system status
End Sub

Sub OnPeriodStart(time)
   FingerprInted = 0
End Sub

Sub OnAvailabilityEvent(eventDetails)
   ' In case it's a failure and within the timeslot then it is counted
```

```
   If context.IsWithinTimeSlot and SystemStatus=UP and _
         eventDetails("Status")=DOWN Then
      FingerprInted = FingerprInted + 1
   End If
   ' update the system status for next comparison
   SystemStatus = eventDetails("Status")
End Sub

Function Result
   Result = FingerprInted
End Function
```

Case Study 13: Dynamic Component Group Handling

It is often required to store values for a group of resources where the members of this group may be dynamic and change during the calculation period. In the following requirement example calculation, it is necessary to perform an intermediate calculation on each of the resources in order to reach the final aggregate result.

Following are a few such examples:

- **Incidents at sites**-calculate the percentage of sites with incidents that took longer than X time to resolve, or count sites that had incidents with resolution time averages that were greater then X.

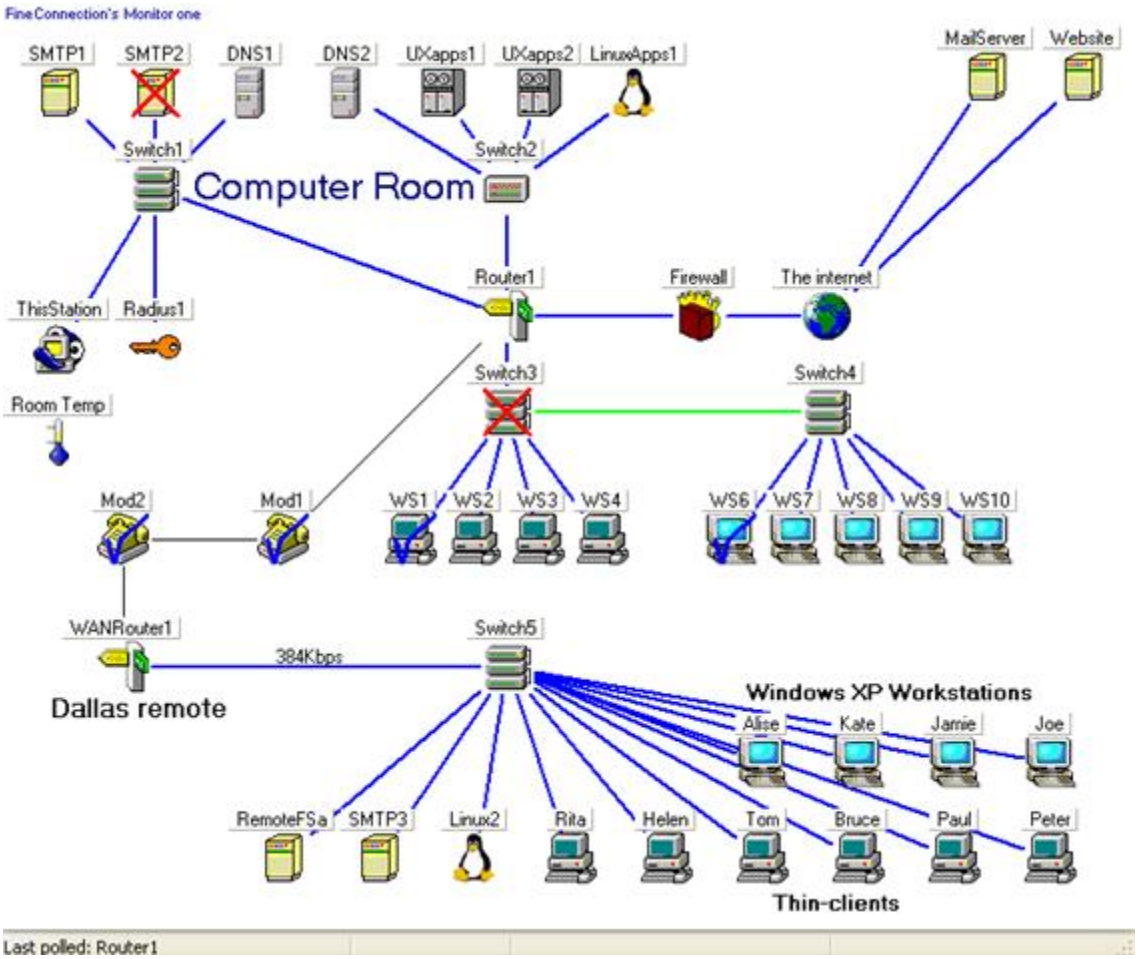
  In these examples, Sites are resources that have incidents associated with them.

- **Server availability**-count the number of servers whose availability time was greater then X%.

  A server is a resource for which availability percentage must be evaluated.

- **Transaction types**-calculate the percentage of transactions types that failed more then X times.

  In this case, a transaction type is a resource that has failure events associated with it. For each transaction type, a failure counter is stored as an intermediate result and the number of different transaction types that had more than X failures are counted.


Fine Connection's Monitor one
SMTP1
SMTP2
DNS1
DNS2
UXapps1
UXapps2
LinuxApps1
MailServer
Website
Switch1
Switch2
Computer Room
ThisStation
Radius1
Router1
Firewall
The internet
Room Temp
Switch3
Switch4
Mod2
Mod1
WS1
WS2
WS3
WS4
WS6
WS7
WS8
WS9
WS10
WANRouter1
384Kbps
Switch5
Dallas remote
Windows XP Workstations
Alise
Kate
Jamie
Joe
RemoteFSa
SMTP3
Linux2
Rita
Helen
Tom
Bruce
Paul
Peter
Thin-clients
Last polled: Router1

***Example*:**

For the following calculation requirement:

Calculate the percentage of availability for a system that is comprised of a cluster of servers. The system is considered available only when all underlying servers are available.

The solution design will be implemented as follows:

The system availability is evaluated by the availability of its underlying Cluster resources. In this case, it is required to manage and store the status of all the Clustered resources in order to evaluate the system status at each point in time.

In order to do so, the formula is required to hold a map (ServerAvailabilityIndicators map in the example code below) that has an entry for each of the monitored resources. Since all map objects require a key field to reference the associated value, this map will have the resource indicator as the key (which is the resource ID) and the value will be the component status. Whenever the status of a component changes, the relevant entry in this map should be changed. Upon each availability event the formula will update the status of the relevant resource in the map and reevaluate the system status accordingly.

Since the monitored resources may change (some might be removed and some might be added during the calculation period) this must be managed within the formula by adding a function that identifies the change and updates the monitored component map by adding a new entry for a new component or deleting an entry if a component was removed.

OnRegistration is the event handler that manages a Registration Event which is triggered when a change in the monitored resources occur. Such a change may occur as a result of committing a resource (or change set) that may produce changes in the resources included in the calculation, according to the formula's registration method.

During each registration it is necessary to update the map that holds the resource statuses with any required changes. This means comparing the map used to hold the resource statuses with the map that holds the resources at the time that the registration is run (based on the registration method), and include all of the resources that were added or delete the resources that were removed.

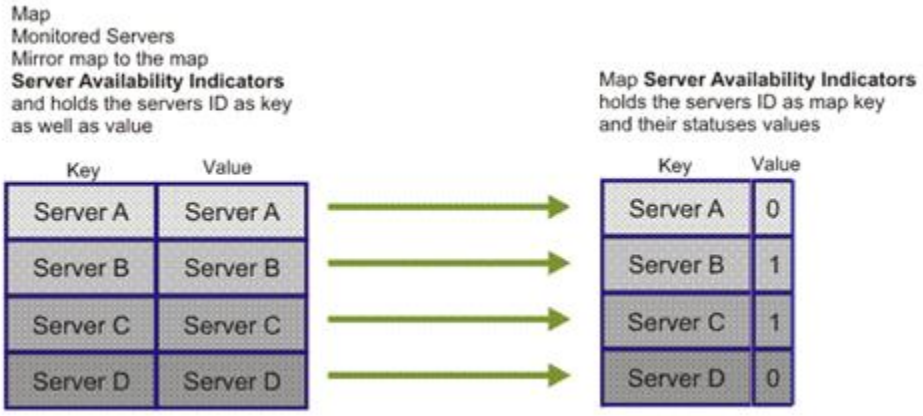
The OnRegistration procedure must therefore execute a function which compares the monitored resources against the new allocated resources in order to structure the monitored resources accordingly.

The Context object has a set of methods which parallel the registration methods. These return a map with all of the resources that are a part of the resources according to the registration method.

In the following example the registration of the formula is ByContractParty and the same method is therefore used by the Context.ResourcesOfContractParty. This returns a map with all of the resources that are a part of this set at the time of registration.

To compare the two maps, it is necessary to iterate through the maps in parallel. Iterating the maps is done using the "For Each" statement. This statement allows iteration through the values of a map and therefore another map is needed as a mirror to the statuses map in order to be able to iterate through the resources and not their status values. This statement iterates the values of a map and not its keys. Therefore, an extra map that holds the IDs both as a key and also as a value is needed. Furthermore, maintain the mirror map to continually mirror the statuses map, so that it always holds the same set of resources. This means that whenever the statuses map is updated, then the mirror map must be updated as well.

The following figure displays the concept of the mirroring maps.



***Example*:**

```
Dim ServerAvailabilityIndicators
Dim MonitoredServers
Set ServerAvailabilityIndicators=CreateObject("SlalomMap.Map")
Set MonitoredServers=CreateObject("SlalomMap.Map")

Sub OnRegistration(dispatcher)
   dispatcher.RegisterByContractParty "OnAvailabilityEvent",_
      "Availability Event", "SAP Production Server"
   Dim AllocatedServers
   Set AllocatedServers = Context.ResourcesOfContractParty("SAP Production Server")
   UpdateMonitoredServers AllocatedServers
End Sub

Sub UpdateMonitoredServers(allocatedServers)
   Dim Server
   For Each Server In allocatedServers
      If Not MonitoredServers.Exist(Server) Then
         MonitoredServers(Server) = Server
         ServerAvailabilityIndicators(Server)=True
      End If
   Next

   For Each Server In MonitoredServers
      If Not allocatedServers.Exist(Server) Then
         MonitoredServers.Erase Server
         ServerAvailabilityIndicators.Erase Server
      End If
   Next
End Sub
```

***Example*:**

The following function demonstrates how the mirror map is used to iterate through the statuses map when required to evaluate the status of the whole system based on the status of each of the monitored resources.

In this example, the system is considered available if all of the resources are available. Having a single component down is enough to consider the system down:

```
Function SystemAvailability
   Dim Server
   For Each Server In MonitoredServers
      If ServerAvailabilityIndicators(Server) = DOWN then
         SystemAvailability=DOWN
         Exit Function
      End if
   Next
End Function
```

A full Business Logic example with dynamic resource handling is described in the following design pattern example.

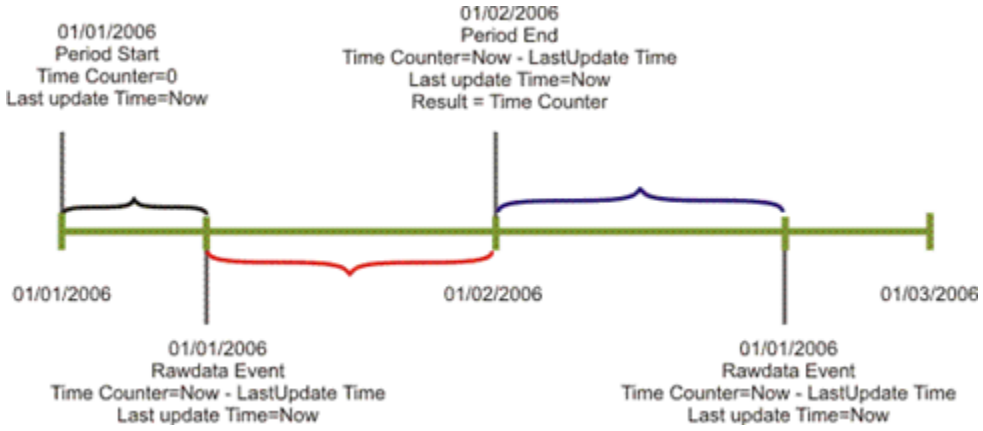
Case Study 14: Time Accumulation Clock Handling

The design pattern described in this section is suitable whenever the required result is a function of a time period that has lapsed between events, for example:

- Available time-calculated as the time passed between one failure and another.
- Resolution time-calculated as the time between the incident opening and the closing time.

For accumulating time, it is necessary to assign a variable in which time can be accumulated (in seconds) and to implement a function that checks both the conditions and time accumulated since the last update time transpired. This function is then executed for every Event received to the formula.

The following figure depicts the clock handling time accumulation.



The LastUpdateTime variable stores the last time the update was executed, regardless whether the time counter was updated or not. The function holds the condition that determines whether the time should be updated and accumulated. For example, the time should not be considered if it exceeds the timeslot period, the system status was Down, or the incident had a pending status.

Although the situation detailed here often uses the Tools.NetTime() function to calculate durations, there may be cases where using the standard VB function DateDiff() is preferable.

The Tools.NetTime function incurs an overhead of checking the timeslot each time it is used. It is recommended to avoid using NetTime in the data event procedures since these procedures are called for any new incoming event and therefore invoke the NetTime call. If your timeslot is 24/7, it is recommended that you use the DateDiff function to avoid the overhead of checking the timeslot.

***Example 1*:**

The following 'update counters' routine accumulates the total period of time in the PeriodNetTime variable. In AvailabilityTime the routine accumulates the time that the system status was Up, meaning that the system was available.

The Tools object contains the NetTime method, NetTime(beginTime, endTime)0. This method returns the number of seconds between beginTime and endTime that are within the timeslot of the current Metric. Any time between these two variables is part of a timeslot that is excluded, hence, it is very commonly used for duration calculations where a timeslot is used. (For example: for Priority 1 Tickets resolved within four business hours, where even though a ticket was raised at the end of a business day and might not be solved until the next morning, it is still within SLA due to the timeslot hours being excluded.)

```
Sub UpdateCounters (time)
   PeriodNetTime = PeriodNetTime + tools.NetTime (LastUpdateTime, time)
   If SystemStatus = UP Then
      AvailabilityTime = AvailabilityTime + tools.NetTime (LastUpdateTime, time)
   End If
   LastUpdateTime = time
End Sub
```

***Example 2*:**

The following example calculates the application availability by handling Up and Down events of several critical components, as well as the maintenance events of those components. If all components are under maintenance, time is not considered as expected availability time.

The subroutine UpdateCounters advances the time counters when necessary and is executed with every event received to the formula (Raw Data Event/Engine Event). It also updates the expected available time in cases where the time is within the timeslot period and the components are not within a planned downtime period. The formula updates the actual availability time only when it also has a system status of "Up".

DateDiff is a standard VB function that returns the time between two dates, but does not exclude any timeslot information.

```
'Force variable declaration
Option Explicit

'Global variables
Dim ExpectedAvailabilityTime
Dim ActualAvailabilityTime
Dim LastUpdateTime
Dim AvailabilityIndicators
Dim MonitoredComponents
Dim DowntimeStatuses

'Map objects creation
Set AvailabilityIndicators=CreateObject("SlalomMap.Map")
Set MonitoredComponents=CreateObject("SlalomMap.Map")
Set DowntimeStatuses=CreateObject("SlalomMap.Map")

'After loading and whenever an infrastructure change occurs
Sub OnRegistration(dispatcher)
   dispatcher.RegisterByResourceGroup "OnComponentDownEvent","Component
Down","Application Components"
   dispatcher.RegisterByResourceGroup "OnComponentUpEvent","Component
Up","Application Components"
   dispatcher.RegisterByResourceGroup "OnMaintenanceStartEvent","Maintenance
Start","Application Components"
   dispatcher.RegisterByResourceGroup "OnMaintenanceEndEvent","Maintenance
End","Application Components"
   UpdateCounters Context.RegistrationTime

   Dim AllocatedComponents
   Set AllocatedComponents = Context.ResourcesOfResourceGroup("Application
Components")

   'make sure formula is monitoring only relevant and all the relevant resources
   UpdateMonitoredComponents AllocatedComponents
End Sub

Sub OnLoad(time)
   'When system goes up for the first time - assume availability OK
   LastUpdateTime = time
End Sub
```

```
Sub OnPeriodStart(time)
   'initializing counters to renew periodic calculation
   ExpectedAvailabilityTime = 0
   ActualAvailabilityTime = 0
End Sub

Sub OnTimeslotEnter(time)
   UpdateCounters time
End Sub

Sub OnTimeslotExit(time)
   UpdateCounters time
End Sub

Sub OnComponentDownEvent(eventDetails)
   UpdateCounters eventDetails.Time
   'write availability status of reported-on resource
   AvailabilityIndicators(eventDetails.ResourceId) = _
      AvailabilityIndicators(eventDetails.ResourceId)+1
End Sub

Sub OnComponentUpEvent(eventDetails)
   UpdateCounters eventDetails.Time
   'write availability status of reported-on resource
   AvailabilityIndicators(eventDetails.ResourceId)= _
      AvailabilityIndicators(eventDetails.ResourceId)-1
End Sub

Sub OnMaintenanceStartEvent(eventDetails)
   UpdateCounters eventDetails.Time
   'write availability status of reported-on resource
   DowntimeStatuses(eventDetails.ResourceId)= _
      DowntimeStatuses(eventDetails.ResourceId)+1
End Sub

Sub OnMaintenanceEndEvent(eventDetails)
   UpdateCounters eventDetails.Time
   'write availability status of reported-on resource
   DowntimeStatuses(eventDetails.ResourceId)= _
      DowntimeStatuses(eventDetails.ResourceId)-1
End Sub

Sub OnPeriodEnd(time,isComplete)
   UpdateCounters time
End Sub

Function Result
   If ExpectedAvailabilityTime <> 0 Then
      Result = 100 * (ActualAvailabilityTime / ExpectedAvailabilityTime)
```

```
   Else
      Result = Null
   End If
End Function

Sub UpdateCounters(time)
   If Context.IsWithinTimeslot And Not AllComponentsAreInPlannedDowntime Then
      'update counter of seconds in period (when availability is expected)
      ExpectedAvailabilityTime = ExpectedAvailabilityTime +
DateDiff("s",LastUpdateTime,time)
      If SystemIsAvailable Then
         'update seconds-of-availability counter
         ActualAvailabilityTime = ActualAvailabilityTime +
DateDiff("s",LastUpdateTime,time)
      End If
   End If
   LastUpdateTime=time
End Sub

Sub UpdateMonitoredComponents(allocatedComponents)
   Dim Component
   'add to monitored Components map all new Components to be monitored
   For Each Component In allocatedComponents
      If Not MonitoredComponents.Exist(Component) Then
         MonitoredComponents(Component) = Component
         AvailabilityIndicators(Component) = 0
         DowntimeStatuses(Component) = 0
      End If
   Next

   'remove from monitored Components map all no-longer-relevant Components
   For Each Component In MonitoredComponents
      If Not allocatedComponents.Exist(Component) Then
         MonitoredComponents.Erase Component
         AvailabilityIndicators.Erase Component
         DowntimeStatuses.Erase Component
      End If
   Next
End Sub

Function SystemIsAvailable
   Dim SystemAvailability
   SystemAvailability = True

   Dim Component
   Dim ComponentAvailability
   For Each Component In MonitoredComponents
      ComponentAvailability = AvailabilityIndicators(Component) = 0 _
         Or DowntimeStatuses(Component) > 0
```

```
      'system availability is evaluated with availability
      SystemAvailability = SystemAvailability And ComponentAvailability
   Next
   SystemIsAvailable = SystemAvailability
End Function

Function AllComponentsAreInPlannedDowntime
   Dim ComponentsInPlannedDowntime
   ComponentsInPlannedDowntime = 0
   Dim Component
   For Each Component In MonitoredComponents
      If DowntimeStatuses(Component) > 0 Then
         ComponentsInPlannedDowntime = ComponentsInPlannedDowntime + 1
      End If
   Next

   If ComponentsInPlannedDowntime = MonitoredComponents.Count Then
      AllComponentsAreInPlannedDowntime = True
   Else
      AllComponentsAreInPlannedDowntime = False
   End If
End Function
```

Writing Effective Business Logic Examples

Best practice recommendations on how to write effective business logic are given on the following subjects:

- **Clustered Metrics**
  - When evaluating the volume of system, count a clustered metric as the number of items in the metric and remember that everything is multiplied.
  - Recalculation of one cluster item means recalculation of the whole cluster. Therefore, remember to consider it when planning the clustering, the way adapters are configured and when changing the resources structure
  - Same Raw Data events that go to many cluster items have high performance cost (context switching)
- **Global Variables**
  - Get parameters and attribute values at each place in the code that they are need. Avoid keeping them in a global variable, especially in clustered metrics (it increases the size of the “states”)
  - Avoid logic that processes big maps. Instead, process each event in the OnXXEvent method
  - Remove items from maps as early as possible. For example, when a ticket is closed and not at period end
- **Design Patterns**

  The Predefined Content package contains several design patterns for common scenarios. Using these design patterns can improve performance
- **Built-in Functionality**

  ACE has built-in functionality and tools for various purposes as follows:
  - Timeslot Functionality
    - NetTime
    - IsWithinTimeslot
  - Time of Events
    - TimeOfLastEvent
    - TimeOfLastEventHandler
  - Context Object
    - Contains many environment-sensitive methods
    - Use these methods and avoid “Safe ODBC”
- **Business Logic Outputs**

Keep the structure in T_SLALOM_OUTPUTS. This means that if you have several logical tables in T_SLALOM_OUTPUTS that have a similar structure, it is very useful to place similar logical fields in the same physical field. This allows easy indexing for improved reports performance

- **Event Reusability**

  Use when:

  - Several metrics are performing first phase calculation, which is by itself needed for the contract, and send events to a summary metric that calculates the result (e.g. financial calculation, high level KPI)
  - A single metric performing a preliminary aggregation on raw data and sends events to several other metrics. Reasonable when the metric sends considerably less events than it gets or performs significant calculations which otherwise are performed many times

  Avoid when:

  - Significantly increasing the number of metrics
  - Implementing more than three levels
  - The complexity of the implementation increases and maintenance becomes harder

- **Recalculations**
  - Avoid massive recalculations as part of the normal operation of the system
  - The reasons for recalculations are:
    - Raw data with a timestamp in the past
    - Event singularity that changes raw data in the past
    - Corrections
    - Exceptions
    - Changes in business logic modules
    - Changes in a contract
    - Event reusability events with a timestamp in the past
    - Changes in the resources structure
  - Consider using the Calculated Data Lock functionality
- **Business Logic Modules**
  - Business logic modules should be written in a way that once fully reviewed, do not need to be changed
  - The guideline is: one module equals one generic logic
  - Business logic modules that are very specific cannot serve many metrics and do not promote code and logic reusability

  - Business logic modules that are too generic are difficult to maintain. In addition, if a business logic module implements many complex logics, a fix in one flow (used by part of the metrics), causes recalculation of all metrics

- **Registration**
  - Perform all filtering of events using registration. Leaving the filtering to the code has a huge performance impact
  - Make it simple
  - For code that is not the registration itself, use the dispatcher.IsRunTimeMode and OnResourceStructureChange methods, especially when there are many resources changes
  - Avoid using the RegisterByEventType method
  - In business logic modules, use a generic form (by contract party, service, resource type) or use parameters or leave the registration to be performed using the user interface (preferred option for Event Reusability)

Case Study 15: Clustered Metrics

Usually, when describing a certain piece of software, the description can be broken into two pieces, the WHAT and the HOW. By WHAT we mean the description of what this piece of code does. The HOW is how does it do it. There is a tendency to concentrate on the WHAT part, and to ignore the HOW part. The reason for that is simple and in many cases justified. By doing so, you reduce the coupling between your components and do not bother your mind with information which is in many cases irrelevant. In many cases tough, the cost of ignoring the HOW part is bad performance.

This case study discusses the way the engine is calculating clustered metrics (answer the HOW part) and describes the performance cost it implies on certain implementations. It also discusses several ways of reducing this cost by changing the implementation.

What are Clustered Metrics

Clustered metrics are metrics that embed in their definition a certain group of resources. This group is referred to as the Cluster of the metric, and each of the resources in that group is referred to as a Cluster Item. When calculating a clustered metric a separate calculation is performed for each of those cluster items. The calculations for each of those cluster items are similar to one another except for:

- Context.ClusterItem – The value of the Context.ClusterItem property which is equal to the cluster item that is being calculated.
- Registrations by Cluster item – When a metric is registered to events by cluster item, each cluster item receives raw data and reusability events that are sent to this cluster item.

How Clustered Metrics are Calculated

The important thing to understand about the calculation of a clustered metric is that all the cluster items are calculated in parallel. By parallel we do not mean that they are calculated by different threads, but that while processing the various events that should be handled by the various cluster items, the events are being processed sequentially and for each event the relevant cluster items are called and they process this event. For example, there are many events that should be handled by many cluster items. There are two ways to do this:

Example: Option 1

```
For each cluster item C
     For each event E that should be handled by C
          Let C handle E
```

Example: Option 2

```
For each event E
     For each cluster item C that should handle E
          Let C handle E
```

The engine handles clustered metrics using the Option 2.

Another important point to understand is that the execution of the VBScript inside the PslWriter is performed by a component called Script Control. There is only one single instance of this component per each metric and this instance is reused for the calculation of the various cluster items. Since the cluster items are calculated in parallel as mentioned before, and since the Script Control component can contain only the data of a single cluster item at each moment, we have to frequently switch the data inside the Script Control component.

To explain this, a more detailed pseudo code that performs the calculation is presented below.

```
1-     For each metric M
2-          Let X be the earliest event not yet handled by M
3-          Let T be the timestamp of the latest state before X
4-       Let L be the list of all events registered by M (all cluster items) starting
from timestamp T until the current time
5-          For each event E in L
6-               For each cluster item C that should handle E
7-                    Let C' be the cluster item that is currently loaded into the
script control
8-                  Take the values of the global variables from the script control
and store them aside for C'
9-                   Take the values of the global variables stored aside for C and
load them into the script control
10-                    Handle event E
```

This whole process of finding some time of an event that was not taken into account yet and then performing calculation from this point onward is called Recalculation. The process of replacing the values of the global variables (steps 8 and 9 in the code above) is called Context Switching.

The two main problems that can be easily seen in the code above are:

- Recalculation is done for all the cluster items together. Since the point in time T (step 3in the code above) is found once and then all the cluster items perform a recalculation based on that point. This means that whenever a single cluster item has some new event, all the cluster items are recalculated.
- Context switching is done very often. This can be easily seen since the context switching (steps 8 and 9 in the code above) are located inside the inner loop.

Recalculation of Clustered Metrics

- Problem Description

  As already explained, all cluster items in a clustered metric are recalculated as a whole. This means that if we have a metric which is clustered over 1000 cluster items and one of them needs a recalculation of the last year due to some new event that arrived, then all the 1000 cluster items are recalculated for the last year.

- Possible Solutions

The following solutions suggestions can reduce the pain of this problem, but they are not always applicable and each has its own disadvantages. The important thing is to understand the problem and its estimated cost and compare this cost to the cost of the proposed solutions.

- When the number of the cluster items is small, we can consider the option of defining each of them as a separate metric. The downside of this approach is of course the maintenance cost of maintaining several metrics as well as the fact that we cannot perform a report for the whole metric and then drill into a specific cluster item
- When the number of cluster items is large, but only one (or few) of them are frequently recalculated, we can consider putting this cluster item in a separate metric and leave all other cluster items in the other metric
- Frequently use a calculation freeze for the relevant contract/metric so that this metric never has long recalculations
- Perform some change in the adapters and the data sources so that there is no long recalculations (i.e. do not send events whose timestamp is older than one month)

Context Switching

- Problem Description

  As already explained, context switching is done in the most inner loop. In other words, for each event that should be handled by each cluster item. When a metric receives many events and when each event is handled by many cluster items, this amount can be very high. Add to this that the context switching operation is relatively expensive (relative to the handling of the event itself in the business logic) and you have a problem.

  The cost of the context switching operation is proportional to the size of the data that is “switched”. The data that we switch during context switch is the values of all the global variables in our business logic (also called “the state”). Thus, the more global variables you have and the larger the size of those global variables is, the more expensive the context switching operation is.

  In particular, it is not recommended to use business logic maps in clustered metrics, especially if the size of those maps can be large.

- Possible Solutions
  - Reduce the time of each of the context switches

    The idea is to reduce the size of the state (global variables). This approach can be done by rewriting the business logic so that it does not contain maps. Of course this is not always possible, but when it is, it is recommended.

  - Reduce the amount of the context switches

    When the cluster is small it is possible to create a separate metric for each cluster item.

Avoid clustered metrics with many clustered items which register to the same events. The idea here is the following:

If each event is handled by a single cluster item then the amount of the context switching is proportional to the number of the events

If each event is handled by all cluster items then the amount of context switching is proportional to the number of the events times the number of the cluster items

Create a non clustered metric that calculates the results for all of the original cluster items (which are now simple resources and not cluster items). Make this metric send the result of each of the cluster items as an event. Create another metric which is clustered and which receives the events from the first metric and report the value received in those events as the result. The idea here is that the large amount of the raw data events will be handled by a non clustered metric and the clustered metric will handle a single event per period per cluster item.

Case Study 16: Business Logic Design Patterns

There are several “Design Patterns” which can be used in many business logics. Those design patterns are tested, and using them when applicable can save a lot of time and in most cases create more efficient business logic. This case study focuses on one such design pattern.

Update Counters Design Pattern

This design pattern is useful, in almost every business logic, which is intended to measure time between certain events. Examples of such business logics are business logics for measuring availability, downtime, mean time between failures, mean time to restore, average response time, average resolution time, percent of components with availability less then X, number of cases not resolved on time, etc.

The common part for all those business logics is that their result depends on the timestamp of various events they receive.

A rule of thumb for deciding if your business logic can benefit from this design pattern would be: if your business logics depends on the timestamp of the various events it receives it most probably should use this design pattern.

Skeleton of this Design Pattern

The code of a business logic that utilizes this pattern can be split into two parts: a framework and an implementation. The framework contains code which in most cases is fixed and does not change for the various business logics. This part is the same for calculation of availability and number of tickets not resolved on time. The implementation contains code which is specific to each business logic.

It is recommended to put those two parts of the code into separate business logic modules and reuse the module of the framework in different metrics.

Here is the code of the framework:

```
Dim g_PrevEventTimestamp

Sub OnLoad(time)
     g_PrevEventTimestamp = time
     InitializeStatusVariables
End Sub

Sub OnRegistration(Dispatcher)
     ' If there is a separate copy of status variables
     ' for each registered resource depend on the registered
     ' resources you should set initial values to the
     ' status variables of the newly added resources here
End Sub

Sub OnPeriodStart(time)
```

```
     InitializeCounters
End Sub

Sub OnPeriodEnd(time, completePeriod)
     HandleEvent (time)
End Sub

Sub OnTimeslotEnter(time)
     HandleEvent (time)
End Sub

Sub OnTimeslotExit(time)
     HandleEvent (time)
End Sub


Function Result()
     Result = CalculateResult()
End Function

Sub HandleEvent(Time)
     Dim diff
     diff = TimeDiff( "s",Time,g_PrevEventTimestamp)
     UpdateCounters(diff)
     g_PrevEventTimestamp = Time
End Sub
```

Here is a skeleton of the implementation module:

```
' Define your status variables here. This can be one
' simple global variable or many complex global variables
' depending on the business logic
Dim g_StatusVar_1, g_StatusVar_2, ... ,g_StatusVar_n

' Define your counters here.
' This can be one simple global variable or many
' complex global variables depending on the business logic
Dim g_Counter_1, g_Counter_2, ... , g_Counter_n

Sub InitializeStatusVariables ()
     ' Set initial values to the various status variables
End Sub

Sub InitializeCounters ()
     ' Set initial values to the various counters
     g_Counter_1 = 0
     g_Counter_2 = 0
     '…
```

```
     g_Counter_n = 0
End Sub

Function CalculateResult ()
     ' Calculate the result. The result should depend on
     ' the values of the counters. It should not depend on
     ' the value of the status variables. It should not
     ' change the values of the counters or the status
     ' variables
End Function

Sub UpdateStatus(method, eventDetails)
     ' Update the value of the status variables based on
     ' the parameters (and posibly on the old value of the
     ' status variables)
End Sub

Sub UpdateCounters(diff)
     ' Update the values of the counters based on their
     ' previous value, on the value of the status variables
     ' and on the value of the diff parameter.
     ' In many cases this calculation is also based on the
     ' value of Context.IsWithinTimeslot
End Sub

Sub OnEvent_1(eventDetails)
     HandleEvent (eventDetails.Time)
     UpdateStatus("OnEvent_1",eventDetails)
End Sub

Sub OnEvent_2(eventDetails)
     HandleEvent (eventDetails.Time)
     UpdateStatus("OnEvent_2",eventDetails)
End Sub

'...

Sub OnEvent_n(eventDetails)
     HandleEvent (eventDetails.Time)
     UpdateStatus("OnEvent_n",eventDetails)
End Sub
```

To better explain this design pattern, here is an example of the implementation of this pattern for calculating availability. This example assumes there are events for UP and DOWN that are handled by two separate event handlers in the business logic. The availability is defined as the percent of time during which the system is up from the total time in the timeslot. The status of the system is assumed to be the status of the last event received (UP or DOWN).

Here is the code of the implementation (the code of the framework is not changed):

```
' Status variable
Dim g_Status
' Counters.
Dim g_UpTime, g_TotalTime

Sub InitializeStatusVariables ()
     G_Status = “UP”
End Sub

Sub InitializeCounters ()
     g_UpTime = 0
     g_TotalTime = 0
End Sub

Function CalculateResult ()
     If g_TotalTime = 0 Then
          CalculateResult = Null
     Else
          CalculateResult = g_UpTime/g_TotalTime*100
     End If
End Function

Sub UpdateStatus(method, eventDetails)
     If method = “OnUP” Then
          G_Status = “UP”
     Else
          G_Status = “DOWN”
     End If
End Sub

Sub UpdateCounters(diff)
     If Context.IsWithinTimeslot Then
          G_TotalTime = g_TotalTime + diff
          If g_Status = “UP” Then
               G_UpTime = g_UpTime + diff
          End If
     End If
End Sub

Sub OnUp(eventDetails)
     HandleEvent (eventDetails.Time)
     UpdateStatus(“OnUp”,eventDetails)
End Sub

Sub OnDown(eventDetails)
     HandleEvent (eventDetails.Time)
     UpdateStatus(“OnDown”,eventDetails)
End Sub
```

There are several variations on this pattern. One of the most common variations is when a separate time counter should be maintained for different entities. For example, when we measure solution time, a separate counter should be maintained for each open ticket. In this case when handling an event which is relevant only to one ticket it is more efficient to update only the counter of that ticket. When a common event is handled (such as OnPeriodEnd or OnTimeslotEnter) the counters of all tickets should be updated.

Note: This variation of the pattern requires maintaining a separate copy of the g_PrevEventTimestamp global variable for each ticket.

Some good examples of the usage of this pattern can be seen in our predefined content. Keep in mind though that this pattern is used a bit differently in the predefined content and the separation between the framework and the implementation is not so obvious there.

Case Study 17: Built-in Functionality

ACE has built-in functionality and tools for various purposes. Using this built-in functionality is preferable to writing it in VBS. Since VBS is an interpreted language, reproducing it in VBS damages performance.

Here is a list of the built-in functions and the appropriate way to use them:

IsWithinTimeslot

This is the simplest of the built-in functions. Its purpose is to enable the business logic to tell whether the system is currently within a timeslot or not. This removes the need to manage a variable in the timeslot enter and timeslot exit functions in order to do the same thing. For example, instead of running the following code:

```
Dim amIWithinATimeslot
Sub OnTimeslotEnter(time)
     amIWithinATimeslot = 1
End sub
Sub OnTimeslotExit(time)
     amIWithinATimeslot = 0
End sub
Sub OnEvent(eventDetails)
     If amIWithinATimeslot = 1 Then
           count = count + 1
     End if
End sub
```

You can run this much simpler code, instead:

```
Sub OnEvent(eventDetails)
     If context.IsWithinTimeslot Then
          count = count + 1
     End if
End sub
```

If you want to use or keep information about the timestamp of the timeslot enter and exit, this functionality would not cover your needs. But normally this is not needed, and this code is sufficient.

TimeOfLastEvent

This function gives you the timestamp of the last raw data or intermediate data event that was handled. This means you do not need to save this information in the event handler, as it is directly available through this function. For example:

```
Function result
     Dim LastEventTimestamp
     LastEventTimestamp = Context.TimeOfLastEvent
End function
```

TimeOfLastEventHandler

This function returns the timestamp of the last event handler called by ace. This includes not only raw and intermediate data event handlers, but also any system events that were called as well. This is especially useful in event handlers that do not receive the time to e.g. the result function. For example:

```
Function result
     Dim LastEventHandlerTimestamp
     LastEventHandlerTimestamp= Context.TimeOfLastEventHandler
End function
```

NetTime

This function allows you to specify two timestamps and to receive the net time (in seconds) that the system was within timeslot for the current rule, between those two timestamps. This especially is a cumbersome functionality and should not be implemented in VBS. Implementing this in VBS would entail keeping a list of all the timeslot enters and exits or calculating the difference between each time of entering timeslot exit directly, in order to figure out the time span between them. Under extreme conditions, this might happen a large number of times and this would not be good for calculation performance. The internal function does the same after significant optimization, and so does it much efficiently. For example:

```
Function result
     Dim MyNetTime
     MyNetTime = Tools.NetTime(MyBeginTimestamp, MyEndTimestamp)
End function
```

The context object

The context object has a variety of parameters that supply information about:

- The current metric.
- The contract it's in.
- The current calculation state.
- Service domain, Domain category and their related values (e.g. threshold).
- Clustering information – The cluster in general and the specific cluster item being handled.
- Functions that return lists of resources based on user requirements.
- Functions that allow you to convert resource names to resource IDs and back.

Accessing this information directly from the database using Safe ODBC is extremely inefficient and makes no sense as the information is readily available from the context object. If possible, always use the built in functionality is a way to get information.

Case Study 18: Registration

Business logic is often written in a way that keeps a map of the metric's resource structure for use during calculations. Since the resource structure changes over time, such business logic needs to update the structure in the map when the resource structure changes.

The OnRegistration method is called when the resource structure changes, as it is responsible for managing engine behavior that has to do with the changes in the registrations and clustering due to resource structure changes. The fact that this method is called for each resource structure change makes it is a convenient place to update the map mentioned above. However, filling the map is not relevant to the registration process. This means that filling the map detracts from the performance of the OnRegistration function. This is not important during runtime, as it normally does not happen very often. However, the OnRegistration method is also called during the infrastructure processing process of the engine, during which the system figures out whether resource structure changes are relevant to the registration of each specific metric the instance is responsible for. During this process, the OnRegistration method is called for every change in resource structure, even if the structure change is not relevant to the current metric. This means that the method may be called a large number of times per metric.

If such logic is implemented in the OnRegistration method, a small degradation in performance during runtime might become a very significant degradation in performance during infrastructure processing.

To solve this problem, filling maps or any other initialization that needs to be run when a change in resource structure occurs, but is not relevant to the registration, can be done in two ways:

Using the IsRunTimeMode property in the dispatcher object.

This property allows the user to find out if the current run is a calculation run or not, and to encase logic that is not relevant to the registration in an 'if' statement that will ensure that it will run only during runtime.

In the example below, the part marked in blue is the part of the business logic that is relevant to the registration and always needs to run. The part marked in green is not relevant to the registration and can be encased in the new 'If' statement.

```
Sub OnRegistration (dispatcher)
     Dim MyResource
     MyResource = Context.ClusterItem
     Dispatcher.RegisterByResource "OnEvent", "My Event Type", MyResource
     Dim ThisResourceMap
     Set GlobalResourceVector= CreateObject("SlalomVector.Vector")
     Dim resource
     Set ThisResourceMap = Context.ResourcesOfResourceGroup(Context.ClusterItem)
     For Each resource In ThisResourceMap
          GlobalResourceVector.Add resource
```

```
     Next
End Sub
```

This code can be improved by changing it in the following way:

```
Sub OnRegistration (dispatcher)
     Dim MyResource
     MyResource = Context.ClusterItem
     Dispatcher.RegisterByResource “OnEvent”, “My Event Type”, MyResource
     If Dispatcher.IsRunTimeMode Then
          Dim ThisResourceMap
          Set GlobalResourceVector= CreateObject("SlalomVector.Vector")
          Dim resource
          ThisResourceMap = Context.ResourcesOfResourceGroup(Context.ClusterItem)
          For Each resource In ThisResourceMap
               GlobalResourceVector.Add resource
          Next
     End If
End Sub
```

Using the OnResourceStructureChanged method.

This method runs right after the OnRegistration method and so gives the same functionality as the original methodology, but it only runs during runtime. This method is not called during infrastructure processing and thus, performance is not damaged.

In the example below, the part marked in blue is the part of the business logic that is relevant to the registration and need to stay in the OnRegistration method. The part marked in green is not relevant to the registration and can be placed in the new function.

```
Sub OnRegistration (dispatcher)
     Dim MyResource
     MyResource = Context.ClusterItem
     Dispatcher.RegisterByResource “OnEvent”, “My Event Type”, MyResource
     Dim ThisResourceMap
     Set GlobalResourceVector= CreateObject("SlalomVector.Vector")
     Dim resource
     Set ThisResourceMap = Context.ResourcesOfResourceGroup(Context.ClusterItem)
     For Each resource In ThisResourceMap
          GlobalResourceVector.Add resource
     Next
End Sub
```

This code can be improved by changing it in the following way:

```
Sub OnRegistration (dispatcher)
     Dim MyResource
     MyResource = Context.ClusterItem
     Dispatcher.RegisterByResource “OnEvent”, “My Event Type”, MyResource
End Sub
```

```
Sub OnResourceStructureChanged(time)
     Dim ThisResourceMap
     Set GlobalResourceVector= CreateObject("SlalomVector.Vector")
     Dim resource
     Set ThisResourceMap = Context.ResourcesOfResourceGroup(Context.ClusterItem)
     For Each resource In ThisResourceMap
          GlobalResourceVector.Add resource
     Next
End Sub
```

Case Study 19: Adapter Wizard for File-based Data Source

This case study examines the best practice methods for integrating with a file-based data source. The example scenario handles a CSV data file that is produced by the source system. With most file-based integrations CA recommends following some basic guidelines, to ensure minimal risk to the integration. These are as follows:

- Where the option is available, we should request that the data is delivered to the CA Business Service Insight Application server's file system. This ensures the delivery mechanism is not dependent upon the adapter trying to fetch data from a remote store (minimizes permission issues with user accounts, and synchronization issues etc.)
- The file naming convention is important since the adapter orders the files according to alphanumeric naming. If we can control this, CA recommends requesting two parts:
  - A sensible naming convention based upon the source file content (especially if more than one file comes from the source)
  - A reverse-order timestamp to ensure the files are ordered with the most-recent file last. (i.e. *<file_name>_yyyymmdd_hhmiss.csv*). The depth of the timestamp used would depend on the frequency of the data delivered

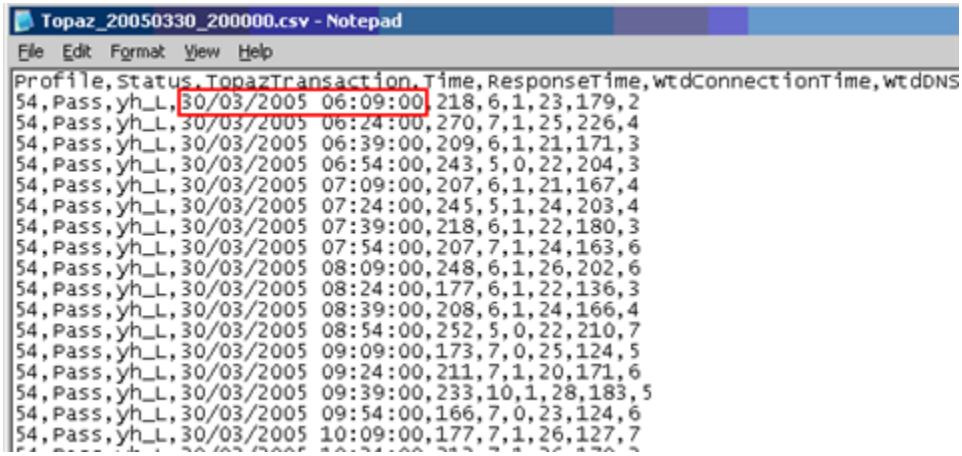
In this scenario, the source files are from a Topaz data source (now known as Mercury Global Monitor, owned by HP). This was created using an API of the product to include the required files for the specific KPIs required. The files are delivered to the CA Business Service Insight Application server directly by an external automated process. The source files are named: *Topaz_yyyymmdd_hhmiss.csv*.

Note: The timestamp of the file is the time at which it was created, hence all entries within the file are leading up to this time.

A sample of the data inside the file can be seen below.

	A	B	C	D	E	F	G	H	I	J	K
1	Profile	Status	TopazTransaction	Time	ResponseTime	WtdConnectionTime	WtdDNSTime	WtdSSLTime	WtdNetworkTime	WtdDownloadTime	
2	54	Pass	yh_L	30/03/2005 06:09	218	6	1	23	179	2	
3	54	Pass	yh_L	30/03/2005 06:24	270	7	1	25	226	4	
4	54	Pass	yh_L	30/03/2005 06:39	209	6	1	21	171	3	
5	54	Pass	yh_L	30/03/2005 06:54	243	5	0	22	204	3	
6	54	Pass	yh_L	30/03/2005 07:09	207	6	1	21	167	4	
7	54	Pass	yh_L	30/03/2005 07:24	245	5	1	24	203	4	
8	54	Pass	yh_L	30/03/2005 07:39	218	6	1	22	180	3	
9	54	Pass	yh_L	30/03/2005 07:54	207	7	1	24	163	6	
10	54	Pass	yh_L	30/03/2005 08:09	248	6	1	26	202	6	
11	54	Pass	yh_L	30/03/2005 08:24	177	6	1	22	136	3	
12	54	Pass	yh_L	30/03/2005 08:39	208	6	1	24	166	4	
13	54	Pass	yh_L	30/03/2005 08:54	252	5	0	22	210	7	
14	54	Pass	yh_L	30/03/2005 09:09	173	7	0	25	124	5	
15	54	Pass	yh_L	30/03/2005 09:24	211	7	1	20	171	6	

Note: CA recommends checking CSV files in Notepad (rather than just Excel) to ensure that the date format is as expected. Excel has a tendency to format the dates according to regional settings of the machine and may not match the actual format seen by the adapter.



Before commencing any adapter creation process, it is also critical that you have done all necessary analysis and investigation into the data source and related KPIs to ensure that the following are known:

- What fields are required by the business logic
- What is the format of the dates in the file
- What is the time zone of the date/time values in the source file (these time zones should be created in the system before commencing the Adapter creation process)
- Are there any date fields that may have blank/NULL entries
- What is the behavior of the data source (are all records added, or are some records an update to a previous event)
- What drill-downs are required for the KPIs (this may impact your choice of “resource”)
- What “Event Types” are required for the effective filtering of the events in the business logic

Once these points are known, you can commence the creation process.

Now, we can perform the adapter creation process based upon this scenario, using the Adapter Wizard.

In our scenario we use the “TopazTransaction” as the resource, the “Profile” as the Event Type, and the “Time” field as the timestamp. We also bring the “Status”, “ResponseTime”, and “WtdDownloadTime” fields into the event structure for use with the business logic.

Adapter Creation

First, ensure the system is ready to create the new adapter and deploy it on the server correctly by checking the following services are started on the application server.

- Adapter Deployment service
- Adapter Listener service

Next, navigate to the Adapters page and create a new adapter. You should choose the option Add New > Text File Adapter from the Adapters page.

General Step

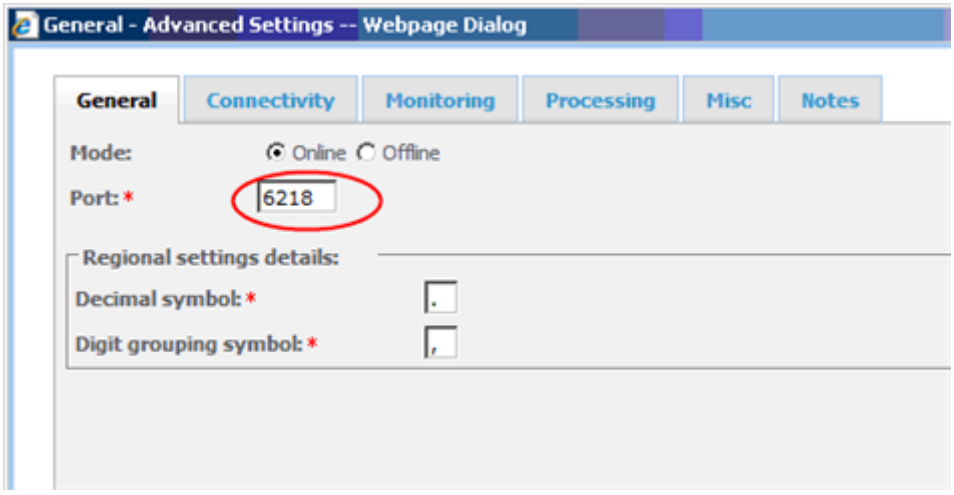
In the General step of the Adapter Wizard enter the following fields:

- *Name*: Provide a suitable name for the adapter
- *Adapter Address*: LOCALHOST is default option (for Application server deployment), but other addresses can be entered using the button alongside if required.
- *Time Format*: This is the default time format used in date/time fields within the data source. Choosing this correctly ensures the fields are automatically detected correctly later in the Wizard process. New Time Formats can be entered here now if required, using the button alongside this field.
- *Time Zone*: This is the time zone that the data records are recorded in. This is required so that the adapter can correctly shift the event_timestamp field (and other date/time fields) back to UTC for correct time-shifting internally. This should have been entered previously as per the data source checklist in the previous section.

Notes:

There is also an “Advanced” button on this page which provides access to a number of configuration parameters for the adapter. Most of these can be left at their default values unless modification is required.

The Adapter “Port” is automatically assigned to the adapter as of CA Business Service Insight onwards, but can be overridden here if required. Other notable parameters that can be changed include the following: Regional settings, Online/Offline mode, Connection details, Monitoring and logging options, Run Once/Always settings, Error limits, file names and comments.



Examining each of these settings is beyond the scope of this case study, but can be found in Adapter Configuration Specifications (see page 311).

Click Next to continue to the next step of the Wizard. The next step provides access to the Data Source Interface of the adapter.

Data Source Interface Step

In the Data Source Interface step of the Adapter Wizard enter the following fields:

- *Data Source Name*: A name for this particular source file (you can have multiple source files in one adapter)
- *File Path*: The path on the Application server (or other server) that contains the source data files. For a server other than the application server, use a UNC notation (i.e. //server01/sourcefolder)
- *Name Pattern*: Use with wildcard character to filter the files located at the "File Path" that are loaded by the adapter.

The Parsing Definition tab also helps to define the structure of the file being imported. The fields can be used as follows:

- *Title*: Boolean values check box to specify whether there is a "title" row in the data file, or not (i.e. the first row in the CSV is a title name, followed by the values on subsequent rows)
- *Delimiters*: Specify the delimiter in the file which separates the individual fields.

Note: There are also two other Advanced buttons here which provide additional configuration options. One button is on the Details tab, and the other is on the Parsing Definition tab. The Advanced button on the Details tab provides access to the following parameters:

Is Active Data Source: Boolean check box that allows you to turn on/off this particular source section of the adapter

After Processing: Allows you to specify whether the file is deleted or kept after processing

Initial File Name: Sets the initial file name to start processing from (in case you do not want to load all files in a particular directory)

Check Data Every: Defines the time interval between checking for new files when the adapter is set to run continuously

The Advanced button on the Parsing Definition tab provides access to define record termination options such as multi-line records etc.

Once the details and parsing definitions are set, it is possible to load a sample of the source file into the Wizard to test the configuration options, and preview the file contents.

By clicking on the Browse button a sample file can be loaded into the preview window below. Navigate to the sample file, and then press the Test File button. This is an optional step.

Note: The Browse function opens from the local machine that you are working on. If this is not the Application server, you must have a copy of the source files that are on the server for this to work. You cannot Browse on the server directly using this function.

Once the file is loaded you should see the contents displayed in the preview window as per below.

Note: The 'Field' name is shown in the header, along with the detected field type (integer, string, time etc) You should check that these were detected correctly and per your requirements before proceeding. Definitely ensure the following:

- Your timestamp was detected as “time” – this is done according to the “Time Format” specified on the first step of the Wizard
- Your resource is detected as a “string”

Once happy with the file preview, click Next. The Mapping step is displayed.

Mapping Step

The next (and Final) step of the Adapter Wizard performs the translation tasks and allows you to map the input fields from the data source, to the output fields, which makes up the CA Business Service Insight “Event”. There are two options to continue with here, depending on whether the “Event Type” is already created in the system or not. There are also a number of other options that you can choose to configure and ensure naming standards are followed. Most of these are optional however, to ease the process and reduce the steps required. Mandatory steps are noted below.

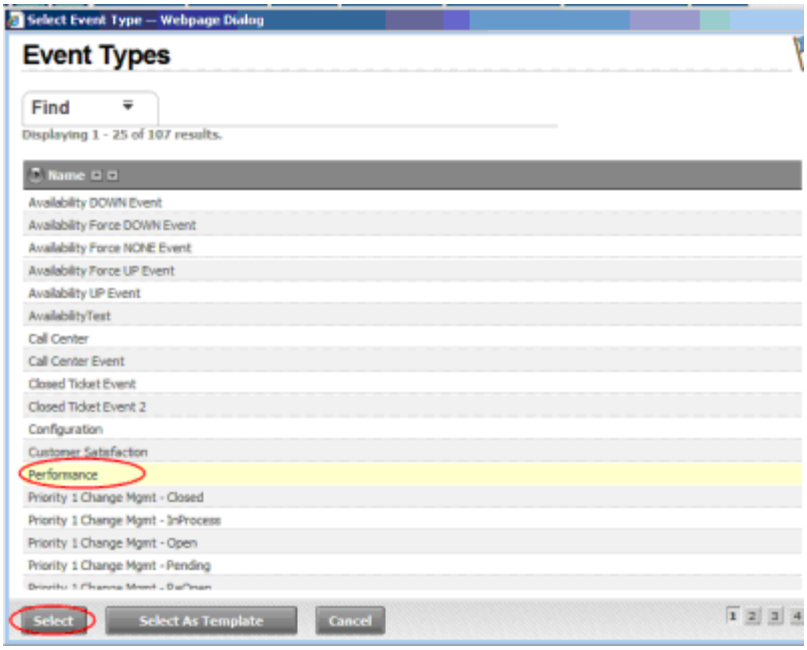
Steps to configure the Mapping step are as follows (includes optional steps):

1. Provide a name for the Input format (useful for adapters with multiple inputs)
2. Add any additional fields required (such as Constant values, Data Source, Title, File Name or Compound field types)
3. Create any required Input Validation criteria
4. Select an existing Event Type, or Create a New Event Type. (mandatory)
5. Perform the mapping of fields from Input to Output (mandatory)
6. Provide a name for the Output Format
7. Perform the mapping for the ResourceId, Timestamp and Event Type
8. Create any required Output Validation criteria
9. Specify the “OnDuplication” setting for the events

When choosing to create a new Event Type, a new window pops up and is pre-populated, based upon the Input Format on the main screen. You still need to enter the Event Type name, and also assign a Resource Type to the Event Type.

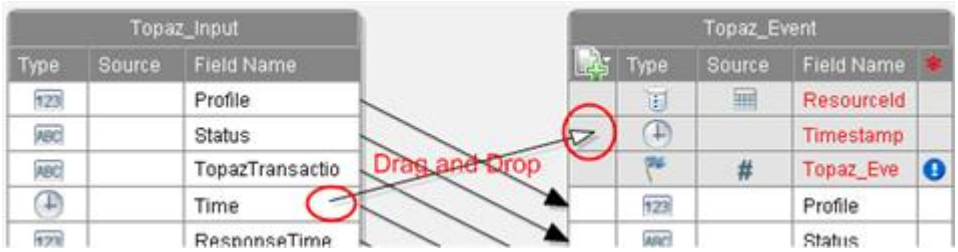
Once complete, you can “Save and Close” and the mapping is completed automatically.

If you choose the “Select Event Type” option, you are presented with a list of the existing Event Types in the system to choose from. When proceeding however, only fields from the Input with matching name and data type from the Event Type are linked automatically. The remaining fields have to be mapped manually.

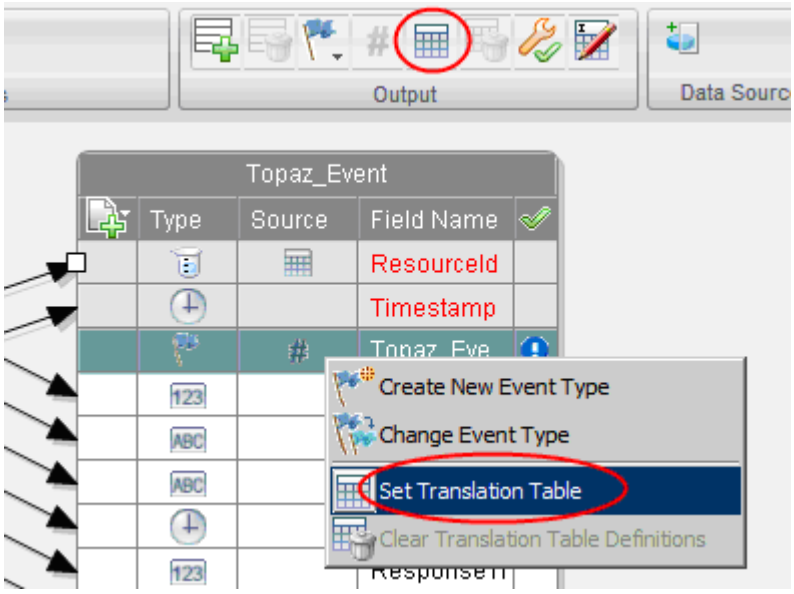


Next step is to configure the ResourceId, Timestamp and Event Type. If the fields exist already (they should be created in previous steps if not) they can be linked to these fields as required.

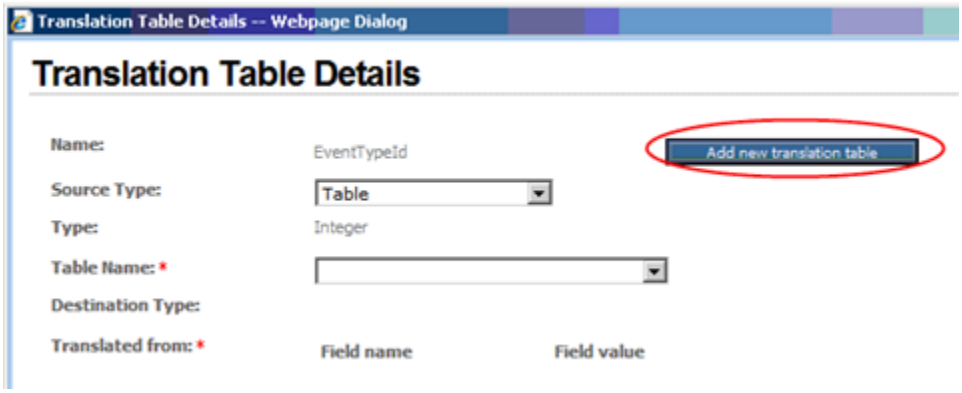
The interface supports a Drag and Drop style of association. If the associated does not persist after setting it, ensure the Type of each one matches. (i.e. Time/String/Integer etc.).



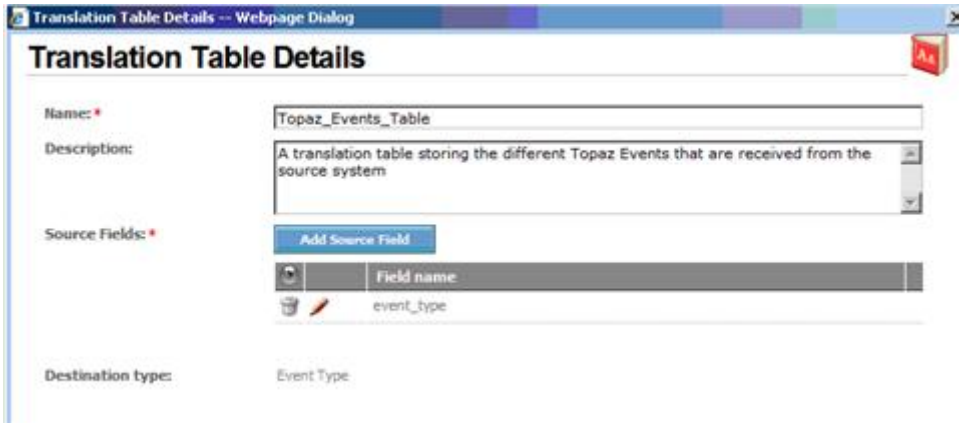
- The ResourceId should be mapped according to requirements (identified during analysis) from a String value in the Input
- The Timestamp should be set to the Time variable that defines the time that the event took place and is to be used for calculation purposes
- The Event Type defaults to a constant value as per the previous screen's “Data Source Name” field. This can be overridden if required. This would be required if multiple events should be sent from this adapter, according to a specific field’s value (i.e. you want to send a different event depending on the value of an Input field). This can be done by right-clicking on the Event Type row (or clicking the button above as indicated in the picture below) and choosing ‘Set Translation Table’



Following this, a pop-up window is presented.



If the translation table for this value field already exists, it can be chosen here, or alternatively a new translation table can be set up for this. The button on the screen above can be used for this purpose.



Once the table is created, you need to specify which of the Input fields is maped to the “Source Fields” specified above.

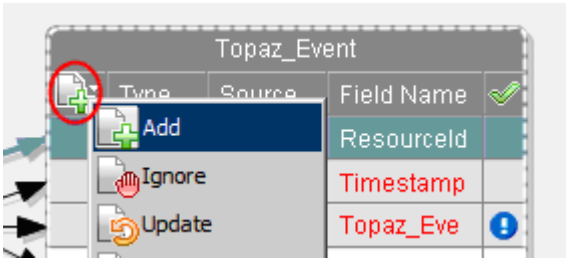
If you wish to specify a different translation table for the resources of the adapter, this should also be done at this point. This is done using a similar process to that described above for the Event Type.

Note: By default, the Adapter Wizard allocated all resources to a pre-existing Translation Table called “Default_Translation_Table” if not otherwise specified. This may be correct for simple implementations but, for more complex implementations (and for data separation purposes) CA recommends using a different table. It is also mandatory when the “Source Fields” of the adapter mapping section are different, or contain greater than one value.

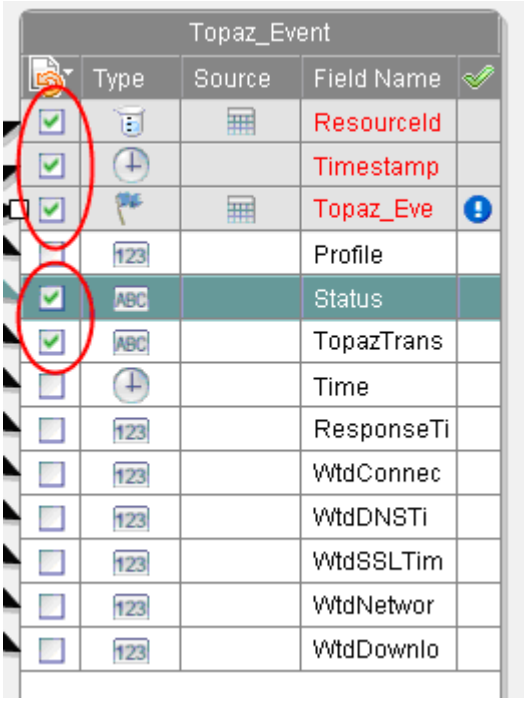
The last step of the Mapping step is to configure the “OnDuplication” setting for the adapter. This setting describes the action taken when a second event, with matching values for all of the “key” fields, is received. This unique “key” can be defined for each Output of the adapter (see further below for more information on this). By default this “OnDuplication” value is set to “Add”, so only needs to be changed if a different action is required. The values available are:

- *Add*: The new event is just added as a separate new event
- *Ignore*: The new event is ignored (dropped) by the adapter
- *Update*: The new event replaces the previously loaded event only if the events “Value” fields are different
- *Update Always*: The new event replaces the previously loaded event

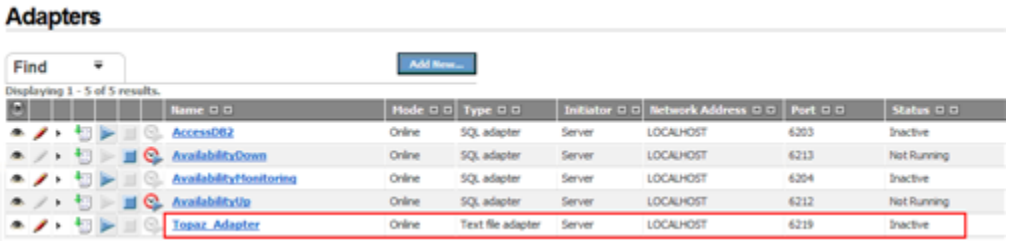
Use of the options other than “Add” may impact the performance of the adapter and should be carefully considered before implementing, especially where data volumes are very large.



If the value is set to anything other than “Add”, the Output structure displays a series of check boxes which are used to define the unique “key”. The “key” consists of each of the items with a check against them. The choice of key should be decided based upon the requirements after careful analysis.



Once the configuration of the mapping section is complete, click the Finish button in the bottom right of the screen. You are returned to the list of adapters in the system, and should now be able to see the adapter you created with a status of “Inactive”.



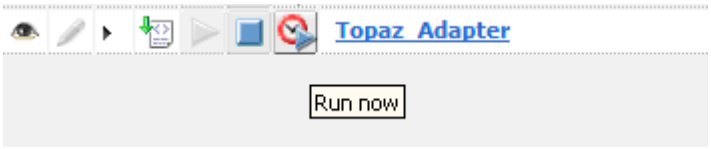
Adapter Deployment

Now that the adapter has been configured, you need to deploy the adapter onto the Application server. Clicking on the “Start Adapter” button, causes the Adapter Deployment Service to deploy the adapter to the file system. This includes the following:

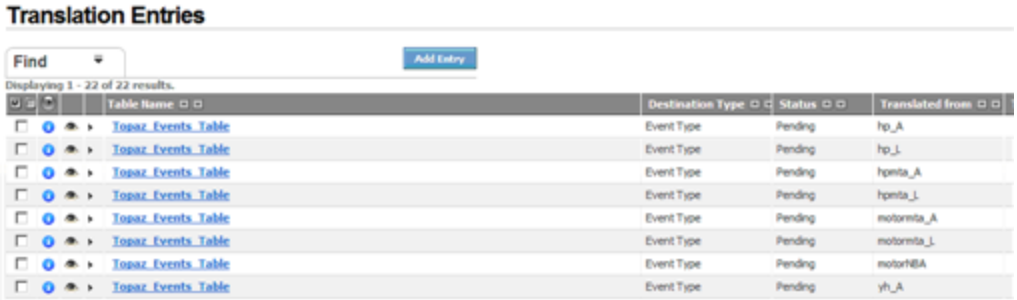
- Creates the folders required to run the adapter
- Copies the XML configuration to the folder
- Creates a shortcut to run the adapter

Once this has been done, your changes are reflected on the server.

From here it is now possible run the adapter from the GUI, by clicking on the “Run Now” button which has become active. This causes the Adapter Deployment service to execute the adapter on the server and commence data collection.



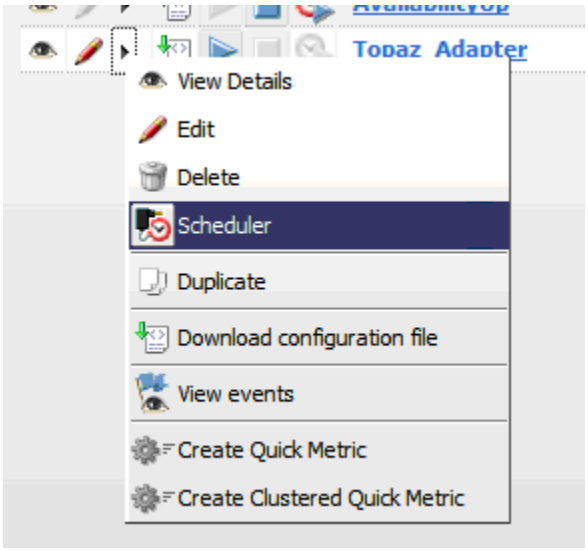
Once the adapter has executed you should now be able to see the Pending Resources and Events under the Pending Translations section of the system.



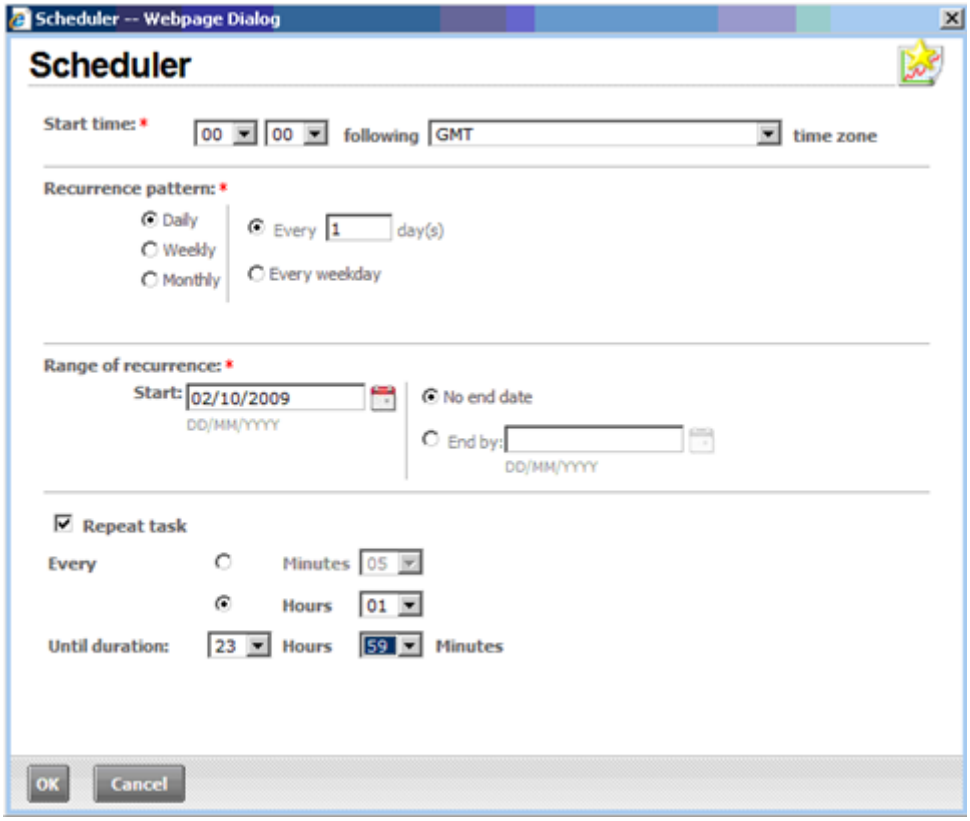
The Pending entries can then be Translated as per normal system configuration. Once Translation are completed the adapter should be run again in order to load the raw data into the system.

Adapter Scheduling

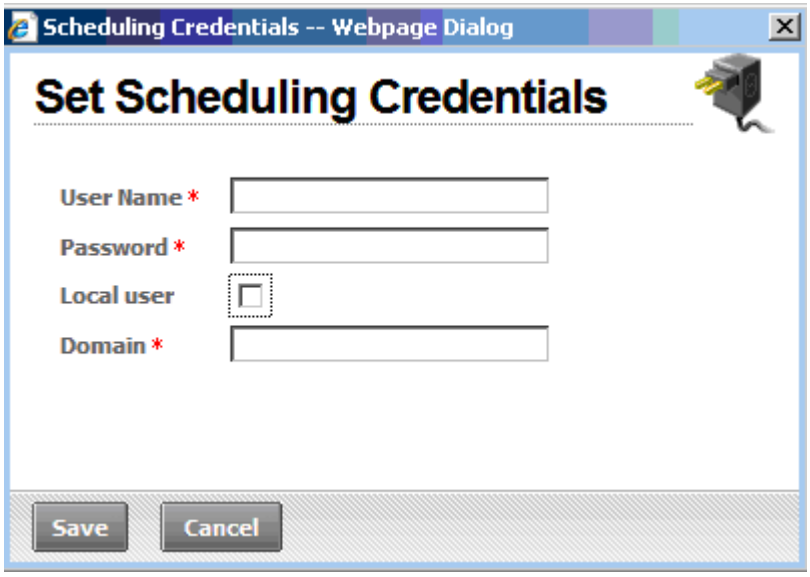
In addition to running the adapter, it is also possible to set a schedule for the adapter from the GUI. The adapter needs to be in the “Stopped” status to do this however. Once stopped you can set the schedule by viewing the context menu for the adapter and choosing “Scheduler”.



The schedule options are the same as those provided by the Windows Task Scheduler. Behind the scenes the Adapter Deployment service actually creates this schedule as an item in the Task Scheduler anyway.



After a schedule is added, and when the adapter is next deployed, the user is prompted to enter the user credentials of the Account on the server that performs the task. This should be entered as the service account created to run the CA Business Service Insight system (default OblicoreSrv), or another administrative account if required.



This integrated scheduler is a very useful feature as it allows the user of the GUI to control the schedule of the adapter without having to directly access the desktop of the Application server (pending appropriate user permissions of course).

Case Study 21: LDAP Integration Example

Organization Requirement

Use the already existing users defined in the organization LDAP server. In addition, the organization portal is used for CA Business Service Insight login and access using the CA Business Service Insight silent login capabilities for Single Sign On (SSO) portals.

Define a Visual Basic (VB) translation script for automatic user creation in the CA Business Service Insight system (LDAP synchronization), the translation script is used to connect to the organization LDAP server and extract the user list from it. CA Business Service Insight tools package methods are used for users, groups and roles creation.

LDAP Connection VB Code Example

```
Option Explicit On
Imports System.DirectoryServices
    Public Function GetLDAPUsers(ByVal ldapServerName As String, ByVal pFindWhat As
String) As ArrayList
        Dim oSearcher As New DirectorySearcher
        Dim oResults As SearchResultCollection
        Dim oResult As SearchResult
        Dim RetArray As New ArrayList
        Dim mCount As Integer
        Dim mIdx As Integer
        Dim mLDAPRecord As String
        Dim ResultFields() As String = {"securityEquals", "cn"}
        Try
            With oSearcher
                .SearchRoot = New DirectoryEntry("LDAP://" & ldapServerName & _
          "/dc=lippogeneral,dc=com")
                .PropertiesToLoad.AddRange(ResultFields)
                .Filter = "cn=" & pFindWhat & "*"
                oResults = .FindAll()
            End With
            mCount = oResults.Count
            If mCount > 0 Then
                For Each oResult In oResults
                    mLDAPRecord =
oResult.GetDirectoryEntry().Properties("cn").Value & "  " &
oResult.GetDirectoryEntry().Properties("mail").Value
                    RetArray.Add(mLDAPRecord)
                Next
            End If
        Catch e As Exception
            MsgBox("Error is " & e.Message)
            Return RetArray
        End Try

        Return RetArray
```

```
    End Function

Sub CheckAddUser
Dim map
Set map = Tools.GetUserDetails("acme@Test")
'Check user already exists
'Tools.AddUserByMap map
'Check with duplicate
map("UserName") = "acme2"
map("UserPassword") = "acme2"
map("UserPasswordExpirationInterval") = "50"
map("UserDescription") = "New description"
map("UserStatus") = "INACTIVE"
Tools.AddUserByMap map
Tools.Commit
End Sub
```

CA Business Service Insight VB Translation Script Methods

- Organization Methods

  AddOrganization / IsOrganizationExists

- Role Methods

  IsRoleExists / SearchRoles

- User Methods

  AddUserByMap / GetUserName

  GetOrganizationName / IsUserExists

  GetUserDetails / SearchUsers

  GetUserFullName / UpdateUserByMap

- User Group Methods

  AddUserGroupByMap / IsUserGroupExists

  DeleteUserGroup / SearchUserGroups

  GetUserGroupDetails / UpdateUserGroupByMap

Create “silent login” code and integrate it into the organization portal to be used for CA Business Service Insight login.

CA Business Service Insight Gatway C# Code example (will be integrated onto organization portal)

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
```

```
using System.Web;
using System.Web.Security;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Security.Cryptography.X509Certificates;
using OblicoreAuthenticationWebService;

namespace Oblicore.SSO
{
      /// <summary>
      /// This sample page is a sample gateway to Oblicore Guarantee(tm) application
interface
      /// The page should be called prior navigating to any page at Oblicore Guarantee
website
      /// or any page using Web Services supplied by Oblicore
      /// The OblicoreGateway page should perform the following actions:
      ///       1) Call Oblicore Authentication Web service in order to authenticate
current user
      ///        2) Call SilentLogin.asp page at Oblicore website to login silently
at Oblicore website
      ///            and create user session context
      ///         3) Redirect to desired page
      /// </summary>
      public partial class _Default : System.Web.UI.Page
      {

            /// <summary>
            /// Oblicore user credentials
            /// </summary>
            struct UserCredentials
            {
                  public string UserName;
                  public string Organization;
            }

            private void Page_Load(object sender, System.EventArgs e)
            {

                  if (Request["OGSESSIONID"]!=null)
                  {
                        //We have been redirected back to this page from
SilentLogin.asp after authentication.
                        //Save OGSESSIONID in cookie for further use
                        HttpCookie SessionCookie = new
HttpCookie("OGSESSIONID",Request["OGSESSIONID"]);
                        Response.Cookies.Add(SessionCookie);
```

```
                        //Redirect to desired page
                        Response.Redirect("/");
                  }
                  else
                  {
                        //First time we enter the page.
                        //Let's perform authentication.

                        string sAuthToken = string.Empty;

                        // Obtain OG user name and organizations from portal user
directory
                        UserCredentials ucOblicoreUser =
GetOblicoreUserCredentials();

                        //Initialize Oblicore Authentication WebServce
                        //Project should include Web Reference to the service
                        //Service is located on Oblicore Guarantee website at
/WebServices/OblicoreAuth.asmx
                        OblicoreAuth oAuthService = new OblicoreAuth();
                     //     oAuthService.ClientCertificates.Add(x509);
                        oAuthService.Url = "https://" + "localhost" +
"/WebServices/OblicoreAuth.asmx";
                        try
                        {
                              //Invoke authentication Web Service.
                              //The AuthenticateUser method returns encrupted
token, which should be passed to
                              //SilentLogin.asp page, located in root folder of
Oblicore Guarantee website
                              sAuthToken =
oAuthService.AuthenticateUser(ucOblicoreUser.UserName,ucOblicoreUser.Organization
);
                        }
                        catch (Exception ex)
                        {
                              //Proceed authentication error if any
                              Response.Write("The error has occurs during Oblicore
authentication: " + ex.Message);
                              Response.End() ;
                        }

                        //Call SilentLogin.asp page along with passing it
authentication folder
                        //SilentLogin.asp page is located Oblicore Guarantee website
root folder
                        //After logging in, SilentLogin.asp page will redirect us
back to the current page along with passing OGSESSIONID parameter
```

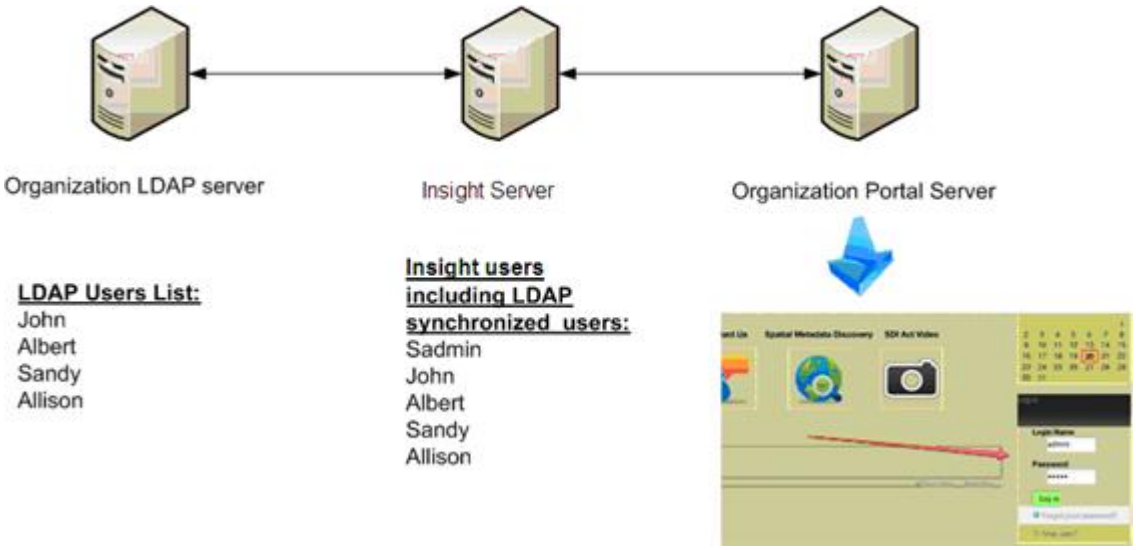
```
//Response.Redirect(ConfigurationSettings.AppSettings["OGURL"].ToString() +
"/SilentLogin.asp?AuthToken="+Server.UrlEncode(sAuthToken)+"&DesiredPage="+GetCur
rentPageURL());

Response.Redirect("https://vit-05/SilentLogin.asp?AuthToken=" +
Server.UrlEncode(sAuthToken) + "&DesiredPage=/Oblicore.asp"); // +
GetCurrentPageURL());
                  }
            }


            /// <summary>
            /// Obtain Oblicore Guarantee user name and organization from portal user
directory
            /// The method is supposed to call ActiveDirectory or another repository
using portal API
            /// to obtain current user name and organization in terms of Oblicore
Guarantee
            /// </summary>
            /// <returns>Oblicore Guarantee user credentials struct</returns>
            private UserCredentials GetOblicoreUserCredentials()
            {
                  UserCredentials ucOblicoreUser = new UserCredentials();
                  //currently alwasy assume user is sadmin and organization is
Oblicore (default)
                  ucOblicoreUser.UserName = "sadmin";
                  ucOblicoreUser.Organization = "Oblicore";
                  return ucOblicoreUser;
            }



            /// <summary>
            /// Retrieves current page URL
            /// </summary>
            /// <returns>Full URL of current page</returns>
            private string GetCurrentPageURL()
            {
                  string s =
(Request.ServerVariables["HTTPS"]==null||Request.ServerVariables["HTTPS"].ToLower
()=="off")?"http://":"https://";
                  s += Request.ServerVariables["SERVER_NAME"] +
Request.ServerVariables["URL"];
                  if (Request.QueryString.ToString() != string.Empty)
                  {
                        s += "?"+Request.QueryString.ToString();
                  }
                  return s;
```

```
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
              //
              // CODEGEN: This call is required by the ASP.NET Web Form Designer.
              //
              InitializeComponent();
              base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
              this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

- Flow Diagram

  The following diagram shows the users sync process and portal access flow. The translation script is configured to run periodically. The script keeps the LDAP user list up to date and adds/removes users as needed.

  The users performs a login to the organization portal. The portal can be configured to redirect them to the CA Business Service Insight server, or display a list of other available applications. The CA Business Service Insight server uses the credentials that were supplied during the initial portal login.


Organization LDAP server
Insight Server
Organization Portal Server
LDAP Users List:
John
Albert
Sandy
Allison
Insight users including LDAP synchronized users:
Sadmin
John
Albert
Sandy
Allison

Case Study 22: Generating Reports using PERL

The following example shows how to use PERL script to connect to the CA Business Service Insight report web service and transferring the criteria xml parameter in a SOAP envelope using an HTTP stream.

Note: The XML code being transferred in the SOAP envelope cannot contain the "<" or ">" symbols, rather the HTML code for these symbols (i.e. <=> >=<)

```
#!/usr/bin/perl

#use strict;

use LWP::UserAgent;
use HTTP::Request::Common;
use XML::Simple;
use Data::Dumper;

my $userAgent = LWP::UserAgent > new(agent => 'Mozilla/5.0');

### Creating a Oblicore Session Context - Oblicore Session ID is stored in $scid ###

my $message = "<?xml version=\"1.0\" encoding=\"utf-8\"?>
<soap:Envelope xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"
xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">
  <soap:Body>
    <CreateSessionContext xmlns=\"http://www.oblicore.com\">
      <userName>sadmin</userName>
      <organizationName>Oblicore</organizationName>
    </CreateSessionContext>
  </soap:Body>
</soap:Envelope>";

my $response = $userAgent > request(POST
'http://obonob/WebServices/OblicoreAuth.asmx',Content_type => 'text/xml', Content
=> $message);

print $response > error_as_HTML unless $response > is_success;

my $result = $response > as_string;

my $xmlstart      = index $result, "<?xml";
my $xmlend       = length $result;

my $xmltext       = substr $result, $xmlstart, ($xmlend - $xmlstart);

my $xml              = new XML::Simple;
my $data            = $xml > XMLin($xmltext);
```

```
my $scid = ($data > {'soap:Body'} > {'CreateSessionContextResponse'} >
{'CreateSessionContextResult'});

print "Session ID : ".$scid."\n";

### Try to get contract list from Oblicore ###

my $criteria_xml =
"&lt;Criteria&gt;&lt;Name&gt;a*&lt;/Name&gt;&lt;Status&gt;EFFECTIVE&lt;/Status&gt
;&lt;/Criteria&gt;";

print  "<?xml version=\"1.0\" encoding=\"utf-8\"?>
<soap:Envelope xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"
xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">
  <soap:Body>
    <GetContractsAdvanced xmlns=\"http://www.oblicore.com\">
      <sessionID>".$scid."</sessionID>
      <criteriaXML>".$criteria_xml."</criteriaXML>
    </GetContractsAdvanced>
  </soap:Body>
</soap:Envelope>";

my $message = "<?xml version=\"1.0\" encoding=\"utf-8\"?>
<soap:Envelope xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"
xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">
  <soap:Body>
    <GetContractsAdvanced xmlns=\"http://www.oblicore.com\">
      <sessionID>".$scid."</sessionID>
      <criteriaXML>".$criteria_xml."</criteriaXML>
    </GetContractsAdvanced>
  </soap:Body>
</soap:Envelope>";

#my $message = "<?xml version=\"1.0\" encoding=\"utf-8\"?>
#<soap:Envelope xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"
xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">
#  <soap:Body>
#    <GetContracts xmlns=\"http://www.oblicore.com\">
#      <sessionID>".$scid."</sessionID>
#    </GetContracts>
#  </soap:Body>
#</soap:Envelope>";

### is_well_formed($message)

print $message;
```

```
my $response = $userAgent > request(POST 'http://obonob/WebServices/Contracts.asmx',
Content_Type => 'text/xml', Content => $message);

print $response > error_as_HTML unless $response > is_success;

my $result = $response > as_string;

print Dumper($result); # Output of contract list

### Close the Oblicore Session ###

my $message = "<?xml version=\"1.0\" encoding=\"utf-8\"?>
<soap:Envelope xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"
xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">
  <soap:Body>
    <ClearSessionContext xmlns=\"http://www.oblicore.com\">
      <sessionContextID>".$scid."</sessionContextID>
    </ClearSessionContext>
  </soap:Body>
</soap:Envelope>";

my $response = $userAgent > request(POST
'http://obonob/WebServices/OblicoreAuth.asmx', Content_Type => 'text/xml', Content
=> $message);

print $response > error_as_HTML unless $response > is_success;
```

Appendix C: Adapter Configuration Specifications

This section contains the following topics:

Overall Structure

The configuration XML file general structure is as follows:

```
< AdapterConfiguration>
    <General…>
    <OblicoreInterface…>
    <DataSourceInterface…>
    <InputFormatCollection…>
    <TranslatorCollection…>
    <TranslationTableCollection…>
</ AdapterConfiguration>
```

Each of the nodes is described in the following sections.

General Section

The General section consists of attributes and sub nodes as following:

XML Structure:

```
<General
MajorVersion="2345"
MinorVersion="01245"
WorkingDirectoryName=" output"
DataSourceControlFileName="DatasourceControl.xml"
SendFileName="Send.txt"
SendControlFileName="SendControl.xml"
LogDebugMode="no"
ConsoleDebugMode="no"
RunOnce="yes"
RepeatUntilDry="yes"
RuntimeErrorLimit="1"
RetryRejectedEvents="yes"
RejectedEventsFileName="rejectedEvents.txt"
RejectedEventsUpperLimit="1000"
RegExUnlimitedSearch (V3.1 Patch)="no"
HoldRejectedEventsSpan="24"
GenerateStatistics="yes"
StatisticsFileName="MyStatistics.txt" >
KeepHistoricState="yes" >
DefaultTimeFormat="%Y/%m/%d-%H:%M:%S"
DefaultDecimalSymbol="."
DataSourceIdMaxLength="60"
DefaultDigitGroupingSymbol=","
SaveIllegalEvents ="no"
WriteEventErrorToLog ="yes"
IllegalEventsDirectoryName="xxxpath"
<DataSourceDifferenceFromUTC
        DefaultOffset="2"
        TimeFormat="%Y/%m/%d-%H:%M:%S">
        <DayLightSaving
                From="2001/04/09-12:00:00"
                To="2001/09/01-12:00:00"
                Shift="1"/>
</ DataSourceDifferenceFromUTC >
</General>
```

- MajorVersion-Specifies the major version number.
- MinorVersion-Specifies the minor version number.
- WorkingDirectoryName-(optional)

  Specifies the default path for Adapter output files (data source control, translation, sends).

Default Value-output directory (folder) in the directory where the configuration file exists.

- DataSourceControlFileName-(optional)

  Name of the file that the Adapter uses to keep track of last data records retrieved from data source.

  Default Value-DataSourcecontrol.xml (If not otherwise specified, the default value is used.)

- SendFileName-(optional)

  Name of the file that holds the events before they are sent to CA Business Service Insight.

  Default Value-Send.txt (If not otherwise specified, the default value is used.)

- SendControlFileName-(optional)

  Name of the file that the adapter uses to keep track of the sending process.

  Default Value-SendControl.xml (If not otherwise specified, the default value is used.)

- DataSourceIdMaxLength-

  Maximum length for the DataSourceId field in t_raw_data table. If the user inserts a string greater than this length, they receive an error in the adapter.

  This value must be less than or equal to the real size of the table.

  Default Value-60

- SaveIllegalEvents-If selected, this attribute indicates that the illegal events must be written to file.

  Default Value-no

- WriteEventErrorToLog-Guides if data errors are written in the T_Log table

  Legal Values-[yes/no]

  Default Value-yes

- IllegalEventsDirectoryName-(no default value)

- SendFileName-(optional)

  Name of the file that holds the records before they are sent to CA Business Service Insight.

  Default Value-Send.txt (If not otherwise specified, the default value is used.)

- §SendControlFileName-(optional)

  Name of the file that the adapter uses to keep track of the sending process.

  Default Value-SendControl.xml (If not otherwise specified, the default value is used.)

- LogDebugMode-(optional)

Legal Values-[yes/no]

When set to yes, the following is sent to log-original row from data source, parsing results, CA Business Service Insight Unified Event.

- ConsoleDebugMode-(optional)

  Legal Values-[yes/no]

  When set to yes, debug messages appear on the consol.

  - **Indicators for reading and translating records**:
    - **.**: For a record that was read and translated to one output event.
    - **i**: For a record that was ignored by the regular expression parser.
    - **I**: For a record that was read and ignored by the translation tables.
    - **R**: For a record that was read but rejected by the translation table (cannot translated by the translation tables).
    - **X**: For a record for which there was a problem when parsing it. It will be ignored and lost or it will be saved in the Illegal Events directory.

  Note: When the read record passes through more than one translator, the record's indication will be enclosed in parentheses. For example: […] or [RRI].

  - **Adapter's status indicators**:
    - **0**: The adapter is alive and does not read any records at the last second.
    - **E**: Error status.
    - **P**: Pause status.
    - **S**: Stop command received from CA Business Service Insight.
    - **B**: Blocked status, the rejected events table is full.
    - **Translation tables indicators**:
    - **L**: Waiting for translation tables.
    - **T**: Translation table sent by/received from CA Business Service Insight.
    - **t**: Changes in translation table sent by/received from CA Business Service Insight.
    - **Connection indicators**:
    - <Connecting CA Business Service Insight: Trying to connect CA Business Service Insight.
    - <Connecting CA Business Service Insight>: Connection established.
    - <Connecting DataSource: Trying to connect Data Source.
    - <Connecting DataSource>: Connection established.
- RunOnce-(optional)

  Legal Values-[yes/no]

When set to no, the adapter will run continuously.

When set to yes, the text file adapter runs, reads records, and stops automatically. The file adapter reads entire files, waits a few seconds, tries to read new records (depending on the SleepTime). If there are no such records, it stops.

An SQL adapter runs each of the queries only once. If RepeatUntilDry is set to no, it stops immediately. If RepeatUntilDry is set to yes, it waits (depending on the SleepTime), tries to run the queries again (depending on the query's SleepTime), and if there are no new records, it stops.

- RepeatUntilDry-(optional)

  Legal Values-[yes/no] Default value-yes

  See the RunOnce attribute above.

- RuntimeErrorsLimit-(optional)

  Determines the errors limit (for example, for errors in the input events) before the adapter is blocked. When equal to 0, the adapter is not blocked.

  Default Value-1 (Meaning that the adapter is blocked after the first error.)

- RetryRejectedEvents-(optional)

  Legal Values-[yes/no] Default value-yes

  When set to yes, records that need translation are kept in the rejected events file, waiting for translation.

  It is not recommended to set this attribute to no because there is a chance that you may lose important data.

- RejectedEventsFileName-(optional)

  Name of the file that the Adapter uses to keep track of data records that were retrieved from the data source and waiting for translation.

  Default Value-rejected.txt (If not otherwise specified, the default value is used.)

- RejectedEventsUpperLimit-(optional)

  Determines the limit of rejected records that are kept in the rejected.txt file. When the adapter reaches this upper limit, it stops and change to block status until these records are translated.

  Default Value-1000

- RegExUnlimitedSearch-Guides the adapter to perform full search according to the regular expression.

  Legal Values-[yes/no]

  Default Value-no

- HoldRejectedEventsSpan-(optional)

Determines number of hours for which to save the rejected events file. If this attribute is not mentioned, the adapter does not erase any events which must be handled before the adapter can start again.

It is not recommended to use this attribute because there is a chance that you may loose important data.

- GenerateStatistics-(optional)

  Legal Values-[yes/no] Default value-yes

  When set to yes, the adapter creates a statistics file containing statistic information every minute.

- StatisticsFileName-(optional)

  Name of the statistics file.

  Default Value-statistics.txt (If not otherwise specified, the default value is used.)

- KeepHistoricState-(optional)

  Legal Values-[yes/no] Default value-no

  When set to yes, the adapter saves all files, except for the log file, to a new directory called Historic state yyyymmdd-hhmmss, where yyyymmdd and hhmmss are the creation date and time formats.

- DefaultTimeFormat-(optional)

  The default time format. If this attribute is specified it is used as the time format in any place where the TimeFormat attribute is omitted. If it is not specified, the TimeFormat attribute in the other elements is mandatory.

- DefaultDecimalSymbol-(optional)

  The default decimal symbol for real fields.

  Default Value-(If not otherwise specified, the default value is used.)

- DefaultDigitGroupingSymbol-(optional)

  The default digit grouping symbol for integer and real fields.

  Default Value-(If not otherwise specified, the default value is used.)

- DataSourceDifferenceFromUTC-Indicates the time difference between UTC and the time zone of the data source carrying machine.

  - DefaultOffset-The offset from UTC when not in daylight saving time.
  - TimeFormat-Specifies the format by which the Daylight Saving details (described next) should be parsed.
  - DayLlightSaving-Specifies one daylight saving period of the data source time zone. This element is optional (meaning, if not selected, there are no daylight saving times) and it can exist more than once. When some elements exist, they must be ordered by time and the periods must not be overlapped.

    - From-Begin date of the period.

- To-End date of the period.
- Shift-Time shift added to the DefaultOffset within daylight saving period.

CA Business Service Insight Interface Section

The CA Business Service Insight Interface section consists of attributes specifying the connection to CA Business Service Insight. There are two modes; online and offline. In online mode the adapter connects to CA Business Service Insight, gets the translation tables and control command from CA Business Service Insight and send events, statues and translation requests to CA Business Service Insight. In offline mode the adapter works with a local translation tables file and writes the events to an output file.

XML Structure for online mode:

```
<OblicoreInterface
        Mode="online">
        <OnlineInterface
                Port="3006"
                ConnectionInitiator="server"
                Address="localhost"
                SecurityLevel="high"
                SecurityKey="12345678901234567890123456789012"
                UseAcknowledgeProtocol="yes"
                PacketSize="50"
                PacketDeadline="60"
                PacketResendTimeout="60"
                WindowSize="10"
      />
</OblicoreInterface>
```

- Port-The TCP/IP port number the Adapter uses to communicate with CA Business Service Insight server.
- ConnectionInitiator-(optional)

  Legal Values-server/adapter. Default value-server.

  Defines the connection initiator, the Adapter side or the Adapter listener on CA Business Service Insight side.

  The initiator plays the 'slave' or client role, while the other is the 'master' or server role.
- Address-Server network address it is mandatory when the initiator is the adapter.
- SecurityLevel-Defines the level of authentication between the Adapter and CA Business Service Insight server ("Hand Shake"). Level options:
  - none-No authentication is performed.
  - high-authentication is performed. The SecurityKey must be supplied.
- SecurityKey-string of 32 digits. It must be the same string as defined to the adapter in CA Business Service Insight database.

  The flow of the "Hand Shake" process is a follows:
  - CA Business Service Insight server connects to the Adapter.

- A random string is sent from the Adapter to CA Business Service Insight server.
- CA Business Service Insight server encrypts the string with a pre-configured key and sends the result back to the Adapter.
- The Adapter encrypts the same random string that was sent to CA Business Service Insight server using the SecurityKey string and compares the results.
- CA Business Service Insight server randomizes a string and sends the string to the Adapter.
- The Adapter encrypts the string with and sends it back to CA Business Service Insight server.
- CA Business Service Insight server encrypts the same string and compares the results to what has been received from the Adapter.
- The connection is established.
- If any fault is detected in any of the phases in the flow, the connection is not established.

- UseAcknowledgeProtocol-(optional)

  Legal Values-[yes/no] Default value-yes

  When set to yes, the adapter uses the acknowledge protocol. When set to no, the adapter sends the messages/packets and does not wait for the acknowledgement from CA Business Service Insight.

  It is not recommended to set this attribute to "no" because there is a chance that you may lose events.

- PacketSize-(optional)

  Legal Values-1 to 1000

  Default Value-50

  Maximum number of events in one packet.

- PacketDeadline-(optional)

  Legal Values-1 to 3600

  Default Value-60

  Number of seconds before sending a packet that is not full.

- PacketResendTimeout-(optional)

  Legal Values-1 to 3600

  Default Value-60

  Amount of time in seconds to wait for acknowledgement message before sending the packet again. This attribute is applicable only when the UseAcknowledgeProtocol attribute is set to yes.

- WindowSize-(optional)

Legal Values-1 to 100

Default Value-10

Number of packets in window. This attribute is applicable only when the UseAcknowledgeProtocol attribute is set to yes.

XML Structure for offline mode:

```
<OblicoreInterface
        Mode="offline">
        <OfflineInterface
                OutputFileName="outputEvents.txt"
                OutputFileType="tsv"
                OutputTimeFormat="%Y/%m/%d %H:%M:%S"
                OutputDecimalSymbol="."
        />
</OblicoreInterface>
```

- OutputFileName-(optional)

  Name of the file to which the adapter writes the output events.

  Default Value-AdapterOutput.txt (If not otherwise specified, the default value is used.)

- OutputFileType-(optional)

  Legal Values-csv/tsv/xml

  Defines the event format.

- OutputTimeFormat-defines the date fields' format. This attribute can be omitted if the DefaultTimeFormat attribute, in the General section, was defined.

- OutputDecimalSymbol-(optional)

  Defines the decimal symbol for read fields.

  Default Value-. (point)

DataSourceInterface Section

The DataSourceInterface section consists of attributes specifying the connection and the connection type between the Adapter and data source (Measurement Tool, CRM, System log, etc.) and is divided into two main types: File Interface and SQL Interface.

File Interface

The File Adapter can be used to retrieve data from log files, scheduled reports or any other text based file, and the DataSourceInterface defines rules of parsing the information from the file data source and extracting it into fields.

The DataSourceInterface section also defines the way the Adapter handles the source file (whether it deletes the original file in case it was created only for the Adapter, or whether it leaves the data alone if it is needed for other uses, and so on).

XML Structure:

```
<DataSourceInterface WorkFileName="MyWorkingFile.txt" >
   <Files>
        <File
            IsActive="yes"
                InputFormat="events"
                Path="D:\adapters\sample_data\"
                NamePattern="adapterXXX*.log"
                DeleteFileAfterProcessing="no"
                Delimiters=","
                IgnoreRedundantDelimiters ="no"
                TitleExists="no"
                SleepTime="10">
        </File>
   </Files>
</DataSourceInterface>
```

- WorkFileName-(optional). When DeleteFileAfterProcessing is set to 'no' - the file is copied to this name, when it is set to 'yes' - the file is renamed to this name. If not specified default value will be taken ('WorkFile.txt').
  - Files-collection of file elements (more then one File can be defined in one Adapter).
  - File-specified the file attributes.
    - IsActive-(optional) [yes/no]. Defines if this file is activate (when set to "no" this file will not be read).
    - InputFormat-the InputFormat associated with this file. The adapter uses the InputFormat to extract the data from the data source.
    - Path-the path to the location of the source data files.
    - NamePattern-sepcifiy the data source file name. Can use wildcard if more than one file uses the same Input Format.

- DeleteFileAfterProcessing [yes|no] - The way the Adapter handles the source file. When the file was created only for the adapter and it can be deleted after processing set it to "yes". The file is renamed, processed, and than deleted.When it set to "no" the file is copied and the processing takes place in the copied file. If new records are appended to the end of this file, the Adapter copies these new records to the work file in the next cycle. If new records are not appended to the file, the Adapter searches for the first file with the same pattern and name greater (in lexicographic order) than the current file. If the Adapter finds such a file, it proceeds to work with this file. The Adapter does not revert to the previous file even if new records are appended to this file. Use "no" when you need to keep the integrity of the source file.
- InitialFileName -the first file name from it the adapter will search for file with the given pattern. Use this attribute when the NamePattern contains wildcards and you do not want that the adapter read old files.

- Delimiters-(optional). One or more characters serving as delimiters, according to which data rows will be parsed into fields, if not specified default value is "\t".
- IgnoreRedundantDelimiters-(optional) [yes/no]. When set to "yes" consecutive delimiters will be treated as one, else a blank field will be created between the delimiters.
- RegExForParser-(optional). A regular expression to use in order to extract fields from the data source replacing the Delimiters specified earlier. For example:

```
<File
    ….
    RegExForParser="^(.{8}) (.{6}) (?:[ ])*([0-9]+) "
/>
```

  See regular expression documentation for more details.

- TitleExists-(optional) [yes/no] Specify if the first non blank line in the data source file is a title line. The title can be used by the Adapter when parsing the data.
- SleepTime-Specify the retrieving data interval (in seconds) i.e. - number of seconds break between the Adapter data retrieval from the source data file.
- LogicLineDefinition-(Optional)

```
<File
    ….
    <LogicLineDefinition
                  FirstLine="Job server:.*"
                  NumberOfLines="5"
    />
/>
```

  In case the dataset is built from the number of lines and not by one line, the following attributes defines the extraction starting point , end point, and the number of lines compromising the data:

  – AllFile-(optional) [yes/no] When set to "yes" all the file considered as one record, one logical line.

- FirstLine-(optional) A regular expression that define the first line of the logical line. It can be specified with/without LastLine and/or NumberOfLines.
- LastLine-(optional) A regular expression that define the last line of the logical line. It can be specified with/without FirstLine and/or NumberOfLines.
- NumberOfLines-(optional) Number of lines in a logical line. It can be specified with/without FirstLine and/or LastLine.
- MatchCase-(optional) [yes/no] Defines whether the regular expression matching is case sensitive.

SQL interface section

SQL Adapter can be used to retrieve data from databases using a SQL statement.

The SQL interface defines the connection to the database and the queries used to retrieve the data, detailed as follow:

XML Structure:

```
        < DataSourceInterface >
                <ConnectionString ConnectionTimeout="60" QueryTimeout="30">
                        <![CDATA[ Driver={Microsoft Access Driver
(*.mDataBase)};DataBaseq=d:\Oblicore\database1.mdatabase; ]]>
                        </ConnectionString>
                <QueryCollection>
                <Query QueryName="cases" InputFormat="cases" SleepTime="3600">
                        <SelectStatement AutoCompleteQuery="yes">
                        select
dateclosed,callid,dateopened,companyname,priority,closedmn,responsemn
                        from calls where dateclosed is not NULL
                        </SelectStatement>
                <QueryKeyFields>
                        <KeyField Name="dateclosed" Sort="asc"/>
                        <KeyField Name="callid" Sort="desc"/>
                        <SelectInitialValues>
                                Select min(dateclosed) , 'min date' from calls
                        </SelectInitialValues>
                </QueryKeyFields>
        </Query>
        <Query QueryName="contracts" InputFormat="contracts" SleepTime="3600">
        <ConnectionString>
                <Segment Type="text"
                Text=" Driver={Microsoft Excel Driver (*.xls)}; DriverId=790;
DataBaseq="/>
                <Segment Type="File">
                <File Path="d:\Oblicore " NamePattern="Availabilty_*.XLS>
                </Segment>
                <Segment Type="text" Text=";"/>
        </ConnectionString>
<SelectStatement AutoCompleteQuery="yes">….</SelectStatement>
<QueryKeyFields>…..</QueryKeyFields>
</Query>
</QueryCollection>
</DataSourceInterface>
```

- ConnectionString-Connection string for accessing the source database.

  The ConnectionString can be defined in the DataSourceInterface element and / or in the Query elements. The ConnectionString definition in the DataSourceInterface element is the default and it is used only in a query without a ConnectionString definition.

The connection string can be defined in one string or by segments. When the ConnectionString element does not contain Segment elements the connection string is taken from the ConnectionString element text. If it contains at least one Segment element, the connection string is concatenated from that.

There are two segment types. The first one is text and contains text that is concatenated to the connection string as is. The second type is file and contains a file name with or without wildcards. The file segment can appear only once and it contains other attributes that define what to do with the read file.

- ConnectionTimeout-(optional). Positive integer number that indicates, in seconds, how long to wait for the connection to open. The value 0 indicates to wait indefinitely until the connection is opened. The default value is 15 seconds.

Note: Some of the providers do not support this functionality.

- QueryTimeout-(optional). Positive integer number that indicates, in seconds, how long to wait for the query to execute. The value 0 indicates to wait indefinitely until the execution is complete. The default value is 30 seconds.

Note: Some of the providers do not support this functionality.

- Segment-specified the Segment attributes.
- Type-(optional) (text, file) Segment type
- Text-the text of the text segment.
- File-specifies the File attributes
- Path-the path to the location of the source data files.
- NamePattern-specifies the source data file name, can use wildcards.
- InitialFileName-(optional). Tells the user from which file to start the search and the pattern to search for.
- UsePath-(optional) [yes, no] When set to yes, the file path is concatenated to the connection string.
- UseFileName-(optional) [yes/no].When set to yes, the file name is concatenated to the connection string (needed for excel files).
- UpdateSchemaFile-(optional) [yes/no]. When set to yes the adapter updates the schema.ini file with the current file name.

Note: Use this attribute only when you want the adapter to change the schema.ini file (database for text files).

- QueryCollection-collection of queries (more than one query can be defined in one Adapter).

- PreliminaryCheck-(optional [yes/no]). The adapter checks the queries when it connects to the database. If this attribute set to "no" the adapter checks the queries by executing them without any change, the adapter runs on the return records later and does not read them again. If it set to "yes" the adapter replaces each "where" string in the select statement with "where 1=2 and" and only than executes the queries. Use this option if you want to check all the queries before the adapter starts to run over the records and when this addition dramatically reduces the query time. Note-Some of the providers do not optimize the query process and when the query is legal the execution takes the same time as it takes without the addition.
- ExternalCommand-(optional). external command that is executed before the adapter starts a new reading cycle.
- Command-name of batch file that is to be executed.
- SleepTime-positive integer number that indicates, in seconds, how long to wait before running the command again.

- Query-specify the query attributes.
  - QueryName-name for the query.
  - IsActive-(optional) [yes/no]. When set to "no" the adapter does not read records from this file.
  - InputFormat-the InputFormat associated with this query. The Adapter uses the InputFormat to extract the data from the source record.
  - SleepTime-time in seconds in which the adapter goes down in order to wait for new data to arrive.
  - Critical-(optional) [yes/no]. If set to "yes" the adapter stops immediately when it finds an error in the query. The use of the option "no" is when it is a known error which is resolved without changing the configuration file.
  - TransactionMode-(optional) [implicit/explicit]. If set to explicit, the adapter initiates a database transaction before the query. This option solves several problems in huge and complicated queries.
  - RollbackSegementName (optional). Defines which rollback segment the adapter uses. Otherwise the database chooses the rollback segment.
- SelectStatement - contains the statement selected to be executed on the source database.
  - AutoCompleteQuery-(Optional) [yes/no]. If set to "yes" the adapter automatically concatenates a where statement to the specified query that does as follows:
  - Creating where statement that will bring only new values based on the key fields.
  - Ordering the results statement based on the key fields.
- QueryKeyFields-defines the fields to start the next data retrieving:

- KeyField:
- Name-the name of the field, taken from the fields of the query.
- Sort-(optional) [asc/desc]. Data sorting method (ascending/ descending).
- Function-(optional). Use this attribute if a special function should be operate on this field, to combine the field value in the function use (:fieldname). For example using the Oracle date function with a field name Ts-Function="to_date(':ts','dd/mm/yyy')"
- ValueLeftWrapper-(optional). Use this attribute to concatenate characters before the value of the field. The default for string and date fields is "'" (apostrophe).
- ValueRightWrapper-(optional). Use this attribute to concatenate characters after the value of the field. The default for string and date fields is "'" (apostrophe).
- NameLeftWrapper-(optional). Use this attribute to concatenate characters before the name of the field. The default is null string.
- NameRightWrapper-(optional). Use this attribute to concatenate characters after the name of the field. The default is null string.
- IsPrimaryKey-(optional) [yes/no]. When set to "no", this key is not used by the adapter in the automatic where statement in the query.
- DefaultInitialValue-(optional). If the SelectInitialValues query does not return records, the adapter takes the default from here. If there are some key fields, all key fields must have this attribute.
- SelectInitialValues-a select statement that provides the initial values for the first where statement (When the control file is empty).

Note: The order of the values must be in the same order of the Field elements for this QueryKeyFields and return a result for each Field.

InputFormatCollection Section

This section specifies the structure of data retrieved from data source-how a data row is to be cut into fields and what are the field types and formats. Initial data filtering and data manipulations may be performed in this section by using InputFormatSwitch and Compound fields respectively.

The general work-flow of this section is as follows:

- Data row is matched against one or more InputFormats.
- Data is dissected into fields following the matching InputFormat specification.
- Compound fields are assigned values by combining and breaking data fields.
- Processed data is checked against TranslatorSwitch conditions.
- Processed data is sent to matching Translator or ignored.

The InputFormatCollection node may contain one or more InputFormat nodes.

XML Structure:

```
<InputFormatCollection>
        <InputFormat InputFormatName="MyInputFormat">
        <InputFormatFields>
                <InputFormatField Name="sid_id" Type="string"/>
                <InputFormatField Name="content" Type="string"/>
                <InputFormatField Name="date" Type="time"
               TimeFormat="%d/%m/%Y %H:%M:%S"/>
                <InputFormatField Name="server" Type="string"
               Source="compound">
                <Compound>
                        <Segment SourceField="content"
                    RegularExpression=".*Job server: ([^\n]+).*" />
                </Compound>
                </InputFormatField>
        </InputFormatFields>
<TranslatorSwitch DefaultTranslator="GeoTranslator">
        <TranslatorCase  TranslatorName="NonGeoTranslator" Break="yes">
                <Condition SourceField="routing_info" Operator="EQ"
            Value="cnano"/>
        </TranslatorCase>
</TranslatorSwitch>
</InputFormat>
</InputFormatCollection>
```

- InputFormat:
  - InputFormatName-Any name for this format, to be referred to by DataSourceInterface section.
- RequiredFields-(optional) Minimum number of fields expected to be found in a data row. A row containing fewer fields is ignored, and an error is logged.

- InputFormatFields-The InputFormatFields may contain one or more Field nodes according to the number of input fields in the data source.
  - InputFormatField-Specifies one data field of the original data row, or a compound field.
    ```
    <InputFormatField Name="timestamp" Type="time"
    TimeFormat="%d/%m/%Y %H:%M:%S"/>
    ```
  - Name-A name for this field, to be referred to by other elements.
  - Type-Data type of the field-string/integer/real/time
  - Source-(optional) Default value taken is "event", possible values:
  - event-Field value is taken from the event coming from data source, values of fields are taken in the same order coming from the data source.
  - compound-Field is a compound. It get its value after making some sort of manipulation on other fields values or constants.
  - title-Field value is taken from title field names. The referred field should be already defined
  - filename-Field value is taken from the data source filename, only for text file adapters.
  - constant-Field value is constant and taken from the ConstantValue property that should appear next.
  - FieldTitleName-When the source is title, specifies the field title to be used. The source field must be defined before.
  - ConstantValue-When source is constant, specifies the value to be matched. When the field's type is time the constant value is formatted string according to the TimeFormat or "Now" or "NowUtc", where "Now" is the current time in the Data source locale and "NowUtc" is the current time in UTC.
  - TimeFormat-For`which the field is parsed. The following character codes may be used for the Date and Time Format:

Sign	Description
%d	Day 1-31
%a	Abbreviated weekday name (sun,mon..)
%A	Full weekday name (Sunday,monday..)
%m	Month 1-12
%b	Abbreviated month name (jan,feb..)
%B	Full month name (january, february..)
%y	Year (2 digits)
%Y	Year (4 digits)
%H	Hour in 24-hour format (00 – 23)
%I	Hour in 12-hour format (01 – 12)
%p	A.M./P.M. indicator for 12-hour clock
%M	Minutes 0-59
%S	Seconds 0-59
%D	Full date without separators (yyyymmdd)
%T	Full time without seconds and separators (hhmm)
%n	Number of seconds from 1970
%E	Date & Time in hexadecimal format.
%%	Percent sign

  - DecimalSymbol - -(optional) The decimal symbol for fields. If not specified, the DefaultDecimalSymbol, from the General section is taken.
  - DigitGroupingSymbol-(optional) The default digit grouping symbol for integer and real fields. If not specified, the DefaultDigitGroupingSymbol, from the General section is taken

- Compound-Required when source=compound. Specifies field manipulations to be collected into a compound field.
  - Segment - Specifies a field manipulation to be added to the created compound. Only the SourceField attribute is required.
  - SourceField-Field to be based on. The referred field should be already defined.
  - RegularExpression-Manipulating regular expression.

- MatchCase-(optional) [yes/no] Define if the regular expression matching is case sensitive.
- SelectionStart-Text extraction start position, starting from 0.
- SelectionLength-Text extraction size.
- Prefix-A string to be prefixed to the manipulation result.
- Suffix-A string to be suffixed to the manipulation result.
- XpathExpression-Manipulating xpath expression.

- InputFormatSwitch-Used to specify format criteria, when data rows come in more than one format.

  Note: When using InputFormatSwitch, the order of InputFormat nodes is important-a referred InputFormat should be already defined.

  DefaultInputFormat-Specifies the name of InputFormat to be routed to, in case no criteria is matched

- InputFormatCase-Specifies a criteria to be tested on data rows to determine to which InputFormat it should be routed.
- InputFormatName-The InputFormat to be used when the criteria is matched.
- LogicOperator-(optional) [and/or].
- and- All conditions must be matched. (the default).
- or- At least one condition should be matched
  - Condition-Condition to be tested on a data row to determine its format.

    SourceField-Field to be tested.

    Operator-Test type, of the following options:

    - **EQ**-Equal to
    - **NE**-Not equal to
    - **GT**-Greater then
    - **LT**-Less then
    - **GE**-Greater or equal to
    - **LE**-Less or equal to
    - **MATCH**-A regular expression should be matched
    - **UNMATCH**-A regular expression should not match

    ValueType-(optional) [constant/field/previousValue]

    - constant-Content of the Value attribute is constant regardless to the source data
    - field-Cntent of the Value attribute is name of field from the same record.

- previousValue-Content of the Value attribute is name of field from the previous record in the same query with the same input format.

Value-Value to be matched, or a regular expression.

MatchCase-(optional) [yes/no] Define if the testing is case sensitive. When set to "yes" the two values translated to lower case before the test.

- TranslatorSwitch-Determines which Translator is to be used to translate the data row into an CA Business Service Insight Unified Event.
  - DefaultTranslator-Translator to be used in case no criteria are matched. If the value is Ignore, no Translator is used and the line is ignored.
  - TranslatorCase-Specifies criteria to be tested on processed data to determine to which Translator it should be routed.

    Break [yes|no]

    yes - (default) If the criteria are matched, do not check next criteria.

    no - At any case, after evaluating the criteria and operating the Translator if matched, roceed to the next criteria.

    LogicOperator-(optional) [and/or].

    - and - All conditions must be matched. (the default).
    - or - At least one condition should be matched.

    TranslatorName-The translator to redirect to if the conditions are met.

    Condition-The condition to be tested on processed data to determine the relevant Translator to be used on it. It is the same as the Condition in the InputFormatSwitch.

TranslationTableCollection Section

The section contains mapping tables from the data source values in to CA Business Service Insight's event fields.

XML Structure:

```
<TranslationTableCollection
        LoadingMode="remote"
        TranslationTablesFileName="Translations.xml">
        <TranslationTable
                Name="ResourcesTranslateTable"
                DestinationType="resource">
                <TranslationField>nodeName</TranslationField>
        </TranslationTable>
        <TranslationTable
                Name="EventTypesTranslateTable"
                DestinationType="event_type">
                <TranslationField>eventType</TranslationField>
        </TranslationTable>
        <TranslationTable
                Name="valueUpDownTranslateTable"
                DestinationType="value"
                ValueType="string">
                <TranslationField>eventType</TranslationField>
        </TranslationTable>
</TranslationTableCollection>
```

- TranslationTablesFileName: (optional) Specify the file name in which the tables are stored locally , if not specified , default value is taken ("Translation.XML").
- LoadingMode: (optional) [standalone, remote].

  Note: The default for online interface is remote and for offline interface is standalone. The specified loading method of the translation tables as following:
- standalone: The Adapter loads the Translation tables locally. No connection with CA Business Service Insight server regarding to the Translation. Changes in the translation tables will be stored only in the local file.
- remote: The Adapter sends a request to load all the tables from the CA Business Service Insight server. Changes made in the Translation tables will be stored also locally.
- LoadTimeout: (optional) When the loading mode is remote you can specify here a timeout in seconds.
  - TranslationTable: Links the event value to the mapping table.
  - Name: A name that will be used and referred by the translator. Legal name starts with letter or underscore and contains letters, digits and underscores."
  - DestinationType: [resource, event_type, contract_party, service, time_zone, value].

- ValueType: (optional) [integer, real, string] The type of the returned value from the table. The string and real values are legal only for table with DestinationType="value"
- TranslationField: The field name to translate from, the name of the field is taken from the input format fields. You can have up to 5 fields.

TranslatorCollection Section

The Translator Collection section describes how to translate the parsed and manipulated data source record extracted in previous section to an CA Business Service Insight event.

When the interface mode is "online" the CA Business Service Insight event has a unified structure that contains the following fields:

- Timestamp: The time of event occurrence.
- ResourceId: CA Business Service Insight's Resource Id associated with the event (the resource that was measured , etc.).
- EventTypeId: CA Business Service Insight's Event type associated with the event and describes the type of the event (type of measurement on the resource , type of ticket action , etc.).
- DataSourceId: (optional) the name of the input data source (file name / table name…).
- Value: The value of the event (measurement result, ticket number, etc.). This field can appear more than one time.

When the interface mode is "offline" there are not any limitations on the number of the fields or on their name.

XML Structure:

```
<TranslatorCollection>
   <Translator TranslatorName="events" OnDuplication = "ignore">
        <TranslatorFields>
                <TranslatorField Name="ResourceId" SourceType="table"
                    SourceName="ResourcesTranslateTable" Iskey="yes"/>
                <TranslatorField Name="EventTypeId"      SourceType="constant"
                    ConstantValue="1002" Iskey="yes"/>
                <TranslatorField Name="Timestamp"        SourceType="field"
                SourceName="timestamp" Iskey="yes"/>
                <TranslatorField Name="Value"    SourceType="table"
                SourceName="valueUpDownTranslateTable" Iskey="yes"/>
                < TranslatorField Name="Value"  SourceType ="field"
                SourceName ="nodeName" Iskey="yes"/>
                < TranslatorField Name="Value"  SourceType ="constant"
                Type="integer" ConstantValue="1000" Iskey="yes"/>
                < TranslatorField Name="Value"  SourceType ="field"
                SourceName ="timestamp" TimeShift="-3600"
                TimeShiftFieldName="createDate"  Iskey="yes"/>
                < TranslatorField Name="Value"  SourceType ="lookup"
                SourceName ="ServiceTable" LookupValue="word"
                Iskey="yes"/>
      </TranslatorFields>
   </Translator>
```

```
</TranslatorCollection>
```

- Translator: Describes how to translate the set of fields it receives into the output event.
  - TranslatorName: Name that is used by the InputFormat to send field sets to this translator.
  - OnDuplication: Member that holds "ignore","add", "update", or "updateAlways" value for determining what to do with duplication event (see Event Singularity)
  - TranslatorFields: Contains list of TranslatorField elements, each one of them contains the following attributes:
  - Name: Field name. In online interface it must be Timestamp, ResourceId, EventTypeId,Value or DataSourceId.
  - SourceType:

    field: Value of this field is taken from field in the input format. The SourceName attribute contains the field name.

    table: Value of the field is taken from the translation table. The SourceName attribute contains the table name.

    lookup: Value of this field is taken from the translation table. The SourceName attribute contains the table name. The value to be translated is taken from the LookupValue attribute and not from the input format.

    constant: Value of the field is constant and its value is in the ConstantValue attribute.
  - SourceName: Contains the field name to translation table name.
  - Type: [integer/real/string/time] Required only when the field's type is not predefined (by field name or by SourceType). In online interface it required only for Value field with SourceType=constant. In offline interface it required for each field with SourceType=constant.
  - IsKey: Represents event's unique key. This key is assembled from fields, which were marked as TranslatorFields?IsKey = "yes".

    (See Event Singularity)
  - LookupValue: Contains the lookup value when SourceType="lookup".
  - ConstantValue: Contains the constant value when SourceType=constant. When the field's type is time the constant value is formatted string according to the TimeFormat or "Now" or "NowUtc", where "Now" is the current time in the Data source locale and "NowUtc" is the current time in UTC.
  - TimeFormat: Contains the TimeFormat, required only for fields with SourceType=constant and Type=time.
  - TimeShift: Defines shift time in seconds, only for time fields.

- TimeShiftFieldName: (optional) Contains field name, from the input format, that contains time shift in seconds. The TimeShift and TimeShiftFieldName can be together.

Appendix D: Defining Business Logic Formulas (Business Logic Expert)

This section contains the following topics:

Things to Avoid When Creating Business Logic Formulas

When creating Business Logic formulas, be sure to bear in mind the following:

- Never assign a null value to a global variable. Assigning a null value may cause the PSLWriter to fail when calculating the Metric. If an uninitialized value is needed, use Empty instead.
- Avoid using map and vector objects in Clustered Metrics. If these objects must be used, keep them as small as possible. Clustered Metrics containing big maps and vectors take a long time to calculate.

Clustered Metrics and resource effectiveness

What is a Clustered Metric?

Clustered Metric enables the definition of one Metric to run for each individual member of a resource group, to apply the same definition and logic for a set of items. A clustering can be set either statically on a predefined set of resources or dynamically on the members of resource group while the group can be changed over time and include or exclude members from the group.

A resource or resource group can be excluded and included in the group over the time and even being excluded from the group and return to become included in the group, over and over again, in the same calculation period (day, month, year etc).

What happens in the Business Logic when a cluster item is removed from the cluster base group?

OnPeriodEnd method and Result Function are triggered for the cluster item. If this happens in the middle of calculation period, the result is only written to the database when the original calculation period is over (e.g. end of the month, end of the year).

What happens in the business logic when a cluster item joins the cluster base group?

Global variables are initiated, OnLoad, OnRegistration and OnPeriosStart methods are triggered for the cluster item

What happens in the business logic when a cluster item joins the cluster base group after it was removed from the group in the same calculation period?

The Result which was set for the period the cluster item was part of the group is overridden with the new Result. In other words, the result in the end of the calculation period refer only the last period in the calculated period where the cluster item is part of the group.

What is the affect of the effective attribute of a resource on the Business Logic?

During the time a resource is not effective, no raw data is collected for the non-effective resource.

What is the affect of the effective attribute of a resource on the clustering?

Changing a resource to become not effective, has the same impact on the clustering like exclusion of the resource from the group. The clustering behaves in the same manner for both resource effectiveness and group membership.

How should you implement exceptions on a resource? Is using resource effectiveness the right method?

There are some business cases where an exception period should be set on an individual resource, for example, a server may get into maintenance and should be neglected from calculations for the maintenance period.

Since Business Logic ignores Raw Data Events of a not-effective resource, you may choose to implement exceptions on a resource using the effectiveness mechanism. It may work for some use cases. However, if the resource is part of a Clustered Metric and the resource is becoming effective and not-effective in the same calculation period, only the last period for which the resource was effective is taken into consideration in the result as stated above. In that case, it is advisable to use the custom attributes feature. An additional attribute for the resource stating the status of the resource can be managed. The Business Logic formula then queries the status of the resource wherever relevant in the script.

Glossary

Adapter

The interface between CA Business Service Insight and third-party data sources. Adapters translate data taken from these data sources into a format that can be used for service level calculations that are provided to contract parties.

Agent

An object that represents a metric in a given time unit.

Alert

Notifications to users about events taking place in the system. Alerts enable users to take corrective action to avoid deviating from contracts, incurring penalties and so on.

Alert Device

Devices in the network that receive alert messages via e-mail, popups or SMS or via a third-party program.

Alert Profile

A set of parameters that defines the conditions under which an alert message is triggered, the users to whom it is issued and how it is sent.

Archived Contract

A contract whose data has been moved to the archive. Archived contracts are not included in calculations and cannot be viewed in reports.

Audit Trail

A chronological record of CA Business Service Insight user and system activities, which are saved in the system. The audit can later be reviewed to identify users actions on the system or processes which occurred on the system.

Booklet

RTF-based file that may include contract data and related reports in a convenient format using pre-defined and configurable design templates

Business Logic Modules

Library of script functions that can be used in Business Logic.

Business Logic Template

Predefined Business Logic formulas that can be used to define new Business Logic formulas.

Change Set

A set of changes to the service provider's resources map.

Clustered Metric

A type of metric that applies to multiple resources or resource groups.

Compound Report

Multiple regular reports that are displayed as multiple series on a single chart

Consumption

A metric type that calculates consumption. Used mainly for the financial module. By using this metric type, it is possible to see in the reports the consumption and the prices separately. The consumption can be calculated in the Price Item metric as well.

Consumption Metric

A metrics that provides the ability to view consumption and prices separately in a report.

Contract Audit Trail

A log of all of the activities of a contract.

Contract Author

The user who is responsible for creating a certain contract.

Contract Party

The customer or provider with whom the contract is signed. A contract party can also represent an internal entity within a larger organization.

Contract Party Group

Logically defined group of contract parties (customers). Referring to a group instead of an individual contract party streamlines contract creation. A contract party can belong to more than one contract party group.

Contract Template

Predefined contract used to create a new contract.

Contract-level Parameter

A parameter of a regular contract, contract template and service level template that is being used by metrics within the business logic.

Current Status Engine

A standalone process that calculates current status metrics. There is no limit to the number of instances allowed to run on one or more machines.

Dashboard

A user interface intended for high-level managers that organizes and presents real-time information and reports in an easy-to-read manner.

Dashboard Map

The layout of the dashboard. The arrangement of widgets in the dashboard.

Data Event

Events generated by CA Business Service Insight adapters

Derived Metric

A metric that was created from a contract template or service level template.

Domain Category

A specific aspect of the service level agreed upon in the contract. For example, you might have a service domain called Help Desk and a domain category such as Response Time of the Help Desk, which describes that particular aspect of Help Desk service. The service provider's system administrator usually defines the domain categories. CA Business Service Insight also supplies predefined domain categories.

Event Annotations

Additional information about a specific event. Event Annotations are set automatically or manually (within the reports).

Event Types

An event is a measurement done on a resource by a third-party measuring tool and then translated by the adapter to a format usable by CA Business Service Insight. Event types determine how these events are defined and formatted for CA Business Service Insight.

Exception

Period of time that does not count towards the calculation of service levels. For example, downtime.

Free-Form Report

A report that is created by a user defined SQL-statement based on CA Business Service Insight database or other external databases.

Granularity

Granularity determines the additional time units the metric author wants to get result for excluding the tracking period of the metric.

Group Report

A report that places several reports side-by-side on one page.

Informational Metric

A metric that performs informational calculation to be used for reporting purposes only.

Interim Metric

A metric that can generate events for the purpose of calculating events. Interim metrics cannot be calculated.

KPI

Key Performance Indicator. A significant measure used on its own, or in combination with other key performance indicators, to monitor how well a business or a service is achieving its quantifiable objectives.

KQI

Key Quality Indicator. A significant measure used on its own, or in combination with other key performance indicators, to monitor how well a business or a service is achieving its quality objectives.

Manual Root-Cause Comment

A comment, set manually by the report viewer, to explain or add additional information to a service level result of specific metric. Manual root-cause comments can be shared by all report viewers of the same metric.

Metric

A combination of several parameters defining the target service level for a certain service at a certain time. Each metric is attached to a single service domain. Metric fields and attributes include domain category, service level formula, service, timeslot, service level target and tracking period.

Metric Event

An event generated by a metric in CA Business Service Insight.

Metric Type

Describes the reason for calculation of a certain metric and defines the list of fields and their behavior in terms of mandatory fields and defaults that a metric of a certain type should have.

Metric Wizard

A user-friendly interface developed to allow users to easily modify metrics the first time the metric is added to the contract from a service level template.

Metric-level Parameters

Parameters of a metric.

Objective Statement

The logical description of a metric containing the parameters that define the metric. Objective statements are displayed in the CA Business Service Insight GUI to provide effective capturing of the metric essence.

OLA

Operational Level Agreement. An OLA is an SLA in which the supplier is the internal organization and the customers are internal business units.

Package

A collection of service level templates and templates packed together to be installed and unpacked in another CA Business Service Insight instance.

Parameter

A value that can be set outside of the business logic to affect the business logic formula. Parameters are being used by the business logic to determine the service level result. A parameter can be of type text, list, number, date or table. Examples of parameters are thresholds and resource names.

Penalty

The compensation owed to the contract party if the service level target is not met within a defined tracking period. Penalties can be turned on or off in CA Business Service Insight.

Price Item

A metric that calculates the prices. The prices can be based on the consumption or defined as fixed prices.

Ratio

A type of Unit.

Raw Data Event

An event generated by raw data in CA Business Service Insight.

Received Events

A list of events received from other metrics shown in the Business Logic Scope window when clicking on the Received Events tab.

Regional Options

Specific details for the locality of the contract party. Parameters include language, currency, time difference from GMT, daylight saving time, date format and currency.

Related Metric

A metric that is referenced as a target in a relation.

Relation

An entity describing a connection of a certain type established between two metrics and having some properties.

Report Wizard

Graphical User Interface (GUI) used to define report parameters.

Report-Group Report

A report that encapsulates multiple regular reports, placed side-by-side

Reports Folder

Reports Folder is a container used to group together reports related to each other in some way or another.

Resource

A single element of the service provider's total infrastructure. Examples of resources include individual servers, switches, hubs, routers, a Help Desk or any other measurable element. Multiple resources can be defined as part of a resource group, which is then treated as another resource itself.

Resource Allocation

Allocating a resource to a service direct the resource events stream to this service. A resource can be allocated to a service, contract party, type or group.

Resource Effective Date

The date from which CA Business Service Insight may use the resource. The resource effective date is defined by the resource version that includes the resource. The resource will not be recognized by any resource version that has an effective date prior to the resource version in which the resource was created.

Resource Entry Date

The date the resource was entered into CA Business Service Insight. This should not be confused with the Resource Effective Date.

Resource Group

A collection of resources that are grouped together as a logical unit. A resource group can contain one or more individual resource or resource group.

Resource Type

A built-in category of resources. Resource must be allocated to at least one resource type.

Role

A set of responsibilities, activities and authorizations. The user role defines the view mode of CA Business Service Insight Every CA Business Service Insight user is assigned one or more roles which determine which actions the user can perform within CA Business Service Insight. Actions that cannot be performed by the role are removed from the CA Business Service Insight user interface when that user accesses the application.

Root-Cause Comment

A comment set within the business logic to explain the service level results. The root-cause comments are associated with metrics.

Service Catalog

A collection of all the information about services, domains and units, templates library and service level templates in CA Business Service Insight.

Service Level Template

A service definition is a predefined set of services and metrics, used to facilitate the creation and modification of contracts according to common business processes.

Simple Report

A means to analyze results calculated by CA Business Service Insight based on a rich set of criteria.

SLO

Service Level Objective. Used to measure contractual agreements.

Tracking Period

The time interval during which CA Business Service Insight measures the provided service level versus a service level objective defined in the contract. The measurement taken during this period determines whether there is a deviation from the defined objective (as defined by the business logic) and whether any penalties or bonuses have been incurred. The tracking period also determines how business reports are generated.

Translation Entry

Represents a definition of source and destination values that are used by the translation table.

Translation Script

A script that can be used to ease the maintenance of translations and prevent errors.

Translation Table

A mechanism used for translating values from the data that comes from the raw data to the data defined in the system.

Translations

The conversion of data collected by adapters from their data sources into entities that have been defined in CA Business Service Insight.

Underpinning Contract

Underpinning contract. A contract with an external supplier covering delivery of services that support the organization in their delivery of services.

Unit

A catalog of measurable units. Examples include percentage and currency.

Unit of Measurement

A unit of measurement for all the metrics defined for a domain category.