

## B.5 Business Relationship

**Role(s):** Service Provider, Customer

**Component(s):** Trade Accounting Service (TAS)  
Billing Calculator (BC)  
Contract Commitment Support (CCS)

**License:** LGPL

### B.5.1 Installation

#### B.5.1.1. Installation Requirements

- Trade Accounting Service (TAS): Java 6 or higher;
- Billing Calculator (BC): Java 5 or higher, Tomcat 5.5 or higher, Apache Axis
- Contract Commitment Support (CCS): .Net Framework 3.5, Windows XP or Windows Server 2003

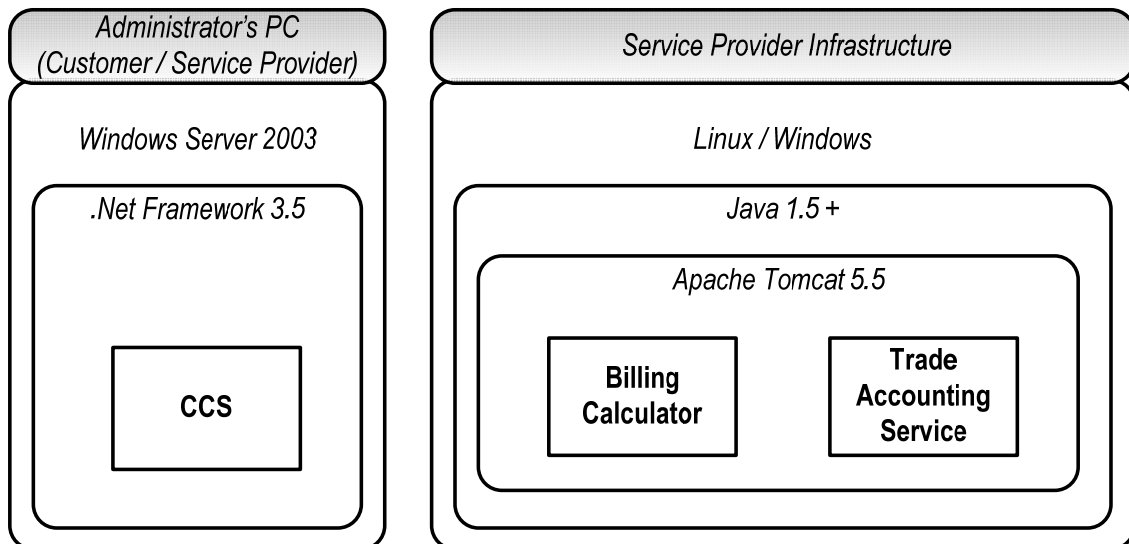


Figure 68: Business Relationship Framework deployment

#### B.5.1.2. Deployment Tips

The Contract Commitment Support components serve two purposes: 1) to invite potential contractors with details about the task & configuration to enact and 2) to configure the infrastructure of both contractor and contracted (or customer and service provider) so as to ensure secure channels, instantiation of relevant resources etc. Accordingly, the two primary concerns for deploying the CCS component consists in 1) accessibility to the internet and 2) accessibility to the infrastructure components and services to be configured. Note that intermediary services may act as a front end to configuration.

It is up to the use case, which communication channels (i.e. invitation and / or configuration) will be secured. Since all communications are web service standard based, WS-Security mechanisms can be easily applied.

## B.5.2 Software Installation

### Trade Account Service (TAS)

#### Prerequisites

The Trade Account Service is a Java application exposed as a Webservice having no particular installation prerequisites, all the libraries that are need should be provided before the installation package:

- Apache Axis
- Apache WSS4J
- GRIA
- Hibernate

It runs with a Java runtime environment version 1.6 or greater.

#### Installation

The TAS component is deployed as a Web service in a `.war` file. By using the BREIN installer, the `.war` file is just deployed inside the AXIS server container. In order to run the application, the user can just link the TAS file which starts AXIS.

To configure the component, the user can update the configuration file (`sess.properties`) with the following properties:

- `log_file`: the path of the file where the logs are stored.
- `log_level`: level of the log (FINE, CONFIG, INFO, WARNING, SEVERE)
- `url_slat`: the URL where the TAS service is deployed

Finally, the services should be accessible locally at the following URL:

<http://localhost:8080/TAS-2.0/TASService>

### Billing Calculator (BC)

#### Prerequisites

The Billing Calculator is a Java set of libraries having no particular installation prerequisites apart from Jena and Pellet (They should be provided before the installation package).

#### Installation

The BC component is deployed as a Web service in a `.war` file. By using the BREIN installer, the `.war` file is just deployed inside the Tomcat server container. In order to run the application, the user can just link the BC file which starts Tomcat.

To configure the component, the user can update the configuration file (`sess.properties`) with the following properties:

- `log_file`: the path of the file where the logs are stored.
- `log_level`: level of the log (FINE, CONFIG, INFO, WARNING, SEVERE)
- `url_slat`: the URL where the BC service is deployed

Finally, the services should be accessible locally at the following URL:

<http://localhost:8080/BC-2.0/BCService>

## Contract Commitment Support (CCS):

### Prerequisites

The Contract Commitment Support needs Windows Server 2003 / XP or higher and the .NET Framework 3.5 or higher.

### Installation

The Contract Commitment Support ships with its own installation file (see below) – however, if the code is altered by the user, we generally recommend using the Visual Studio environment (note that the express editions are free). Nonetheless, all code can also be compiled from the command line tool using MSBuild which ships with the .NET Framework:

```
MSBuild.exe ContractCommitmentSupport.sln
```

To install the CCS using the setup tool, just execute `CCS_Setup.msi` and follow the instructions on the screen. The installer allows you to select whether you want to install the client side, service provider side or both CCS components.

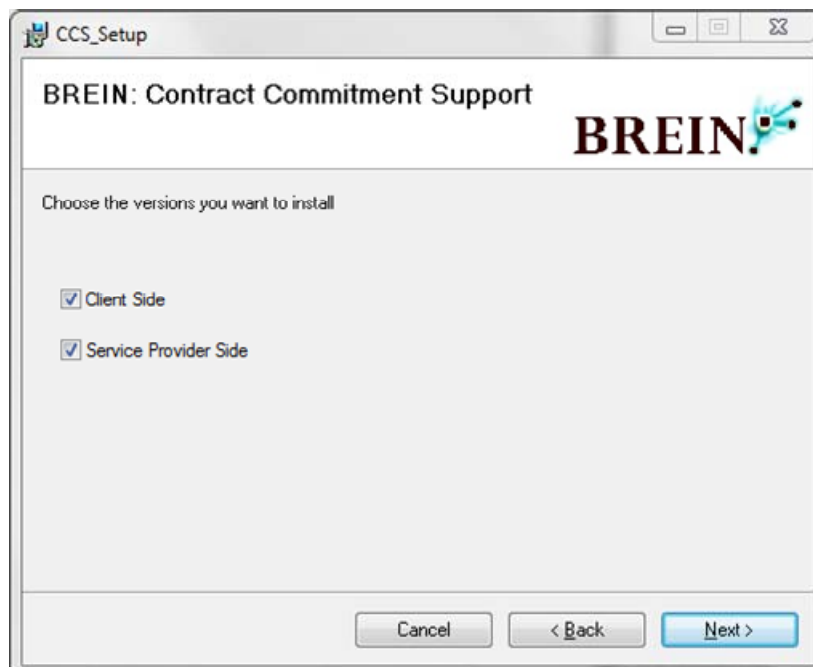


Figure 69: CCS Installer

Note that the client side CCS will, by default, register itself at the URI

- <http://localhost:80/BREIN/CCSClient>

whilst the service provider side CCS will register at

- <http://localhost:80/BREIN/CCSSP>

In addition, the CCS exposes two workflows as web services, which are registered at

- <http://localhost/WWF/ServiceProviderConfigWFSservice>

and

- <http://localhost/WWF/CustomerConfigWFSservice>

respectively.

The Contract Commitment Support is actually an application and not a standalone Web service – accordingly, the application has to be started in order to make the interface available over the Internet.

### B.5.3 Usage Instructions

#### Trade Account Service (TAS)

Here is where the line items that make up the bills to be sent to the customers are stored, together with a record of the bills themselves and customers’ payments against the bills. Different situations need different inputs and outputs.

When a usage on a service is being recorded, the input comes from the billing calculator and consists of the account ID, a description of the usage, who used it and the amount.

The account owner (i.e. the budget holder responsible for paying the bill) can query the Trade Account Service to get a statement of how much they owe. The input to this is the account ID, and the output is an itemised statement of the usage on the account (who used it and what was used) together with the associated cost (this is similar to many types of itemised bills for example for a mobile phone, a restaurant or a hotel).

Finally (and probably most importantly for the provider) is the recording of payments on the account. Payments are handled out of band, as they go through the conventional banking system, but the trade account must be made aware of payments from a customer, and hence there is an interface for logging these payments. The input is the account ID and the amount. The result is that the amount the customer owes the provider is reduced by the amount of the payment and this is shown on the statement.

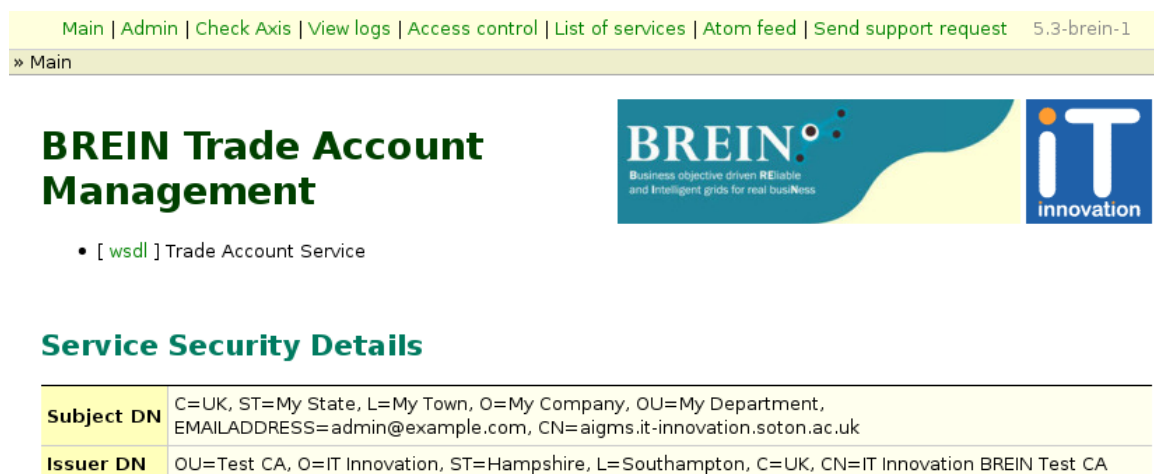


Figure 70: Trade Account Service Web application

#### Billing Calculator (BC)

The billing calculator determines how much to charge a customer based on the SLA they are using, and how much service they used. Hence, the input to this is the usage on the SLA and the SLA ID. The output is a call to the Trade Account Service with the amount to bill the customer and the ID of the account.

This is a service hosted by a service provider and used to keep track of their customers’ spending at the service provider, from which statements (bills) can be

generated. The Billing Calculator connects to this service, and reports the amounts owed by a particular customer. The customer can access this service to find out the amounts they owe, and make payments out of band. The service provider's manager can record the customer's payments.

**Table 21 - Billing Calculator Interfaces**

<b>Method</b>	<code>void configure(Configuration cfg)</code>
<b>Arguments</b>	Configuration cfg – An object that contains all the information needed to change the configuration of the BC and its internal elements.
<b>Return Value[s]</b>	None
<b>Description</b>	Through this method the BC is configured
<b>Method</b>	<code>void notify(Notify notification)</code>
<b>Arguments</b>	Notify notification – A notify object containing notification's topic, message and publisher identifier
<b>Return Value[s]</b>	None
<b>Description</b>	Through this method the BC receives the usage and performance notifications. The notifications follow the Web Service Notifications standard.
<b>Method</b>	<code>List getBillingItems()</code>
<b>Arguments</b>	None
<b>Return Value[s]</b>	List of current Billing Items