

|  |  |
| --- | --- |
|  | **icenter** |
|  |  |
|  | BackgroundSystem 2.0 |
|  |  |
|  | Implementation concept |
|  |  |
|  | Technical Description |
|  |  |
|  | Version: 1.0 |
|  | State: Released |
|  | Classification: Internal use only |
|  | Author: LEF |
|  | Creation date: 2015-01-29 |
|  | Repository: |
|  | Gorba AG  Sandackerstrasse  9245 Oberbüren  Switzerland |

**Table of contents**

[1 BackgroundSystem 2.0: implementation concept 0](#_Toc430339756)

[1.1 Introduction 0](#_Toc430339757)

[2 BackgroundSystem 1.x 0](#_Toc430339758)

[2.1 Overview 0](#_Toc430339759)

[2.2 Managers 0](#_Toc430339760)

[2.3 Issues 1](#_Toc430339761)

[3 BackgroundSystem 2.0 1](#_Toc430339762)

[3.1 Overview 1](#_Toc430339763)

[3.2 Configurational vs Operational data 1](#_Toc430339764)

[3.3 Notifications 2](#_Toc430339765)

[3.4 What’s included in version 2.0 2](#_Toc430339766)

[3.5 Data services 2](#_Toc430339767)

[3.6 Extended data services 2](#_Toc430339768)

[3.7 Updatable data services 2](#_Toc430339769)

[3.8 Functional services 3](#_Toc430339770)

[3.9 BackgroundSystem 2.x 3](#_Toc430339771)

**Modification management**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Dept.** | **Modifications** | **State** |
| 0.1 | 29.01.2015 | LEF | SW | Initial draft | Draft |

**Review**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Dept.** | **Remarks** |
| 0.1 | 18.09.2015 | OCS | SW | Removed all ServiceBus entries |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Release**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Dept.** | **Remarks** |
| 1.0 | 18.09.2015 | OCS | SW | 1st Release |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# BackgroundSystem 2.0: implementation concept

## Introduction

This document describes the concept of the new BackgroundSystem 2.0, highlighting the changes compared to the old version 1.x and the process to migrate components.

# BackgroundSystem 1.x

## Overview

BackgroundSystem 1.x consisted of services providing operations for both data management and functional requests/tracking.

The following services were defined:

|  |  |  |
| --- | --- | --- |
| Name | Description | Main operations |
| AlarmService | Handles alarms sent by unit and confirmed by an operator | CRUD  Confirmation |
| AlertService | Handles alerts generated by applications | Add |
| DomainService | Handles stop points, their associations with units and filters | CRUD (StopPoint specific) |
| ItcsConfigDataService | Provides the configuration for Itcs clients | CRUD (Itcs specific) |
| ItcsService | Provides core functionalities to interact with Itcs clients | Duplex service to get configuration and data |
| ItcsTextMappingService | Provides operations to create | Operations to map texts |
| MaintenanceService | Provides utility operations to maintain the system | Maintenance operations ( example: restart) |
| MembershipService | Handles user and tenant management | CRUD and authentication |
| OperationService | Handles operations | CRUD, manage lifetime of operationjs |
| UnitService | Handles units | CRUD  Restart  TimeSync |

## Managers

“Background” operations were internally handled by managers:

* AlarmManager  
  Handling of alarms. This manager also takes care of acknowledging them sending the ack message through Comm.S
* AlertManager  
  Dispatching alerts
* ItcsManager  
  Storing data table entries and forwarding them to units
* OperationManager  
  Starting/stopping operations
* UnitManager  
  Gateway for unit-related operations (start/stop, realtime, messages, etc.)

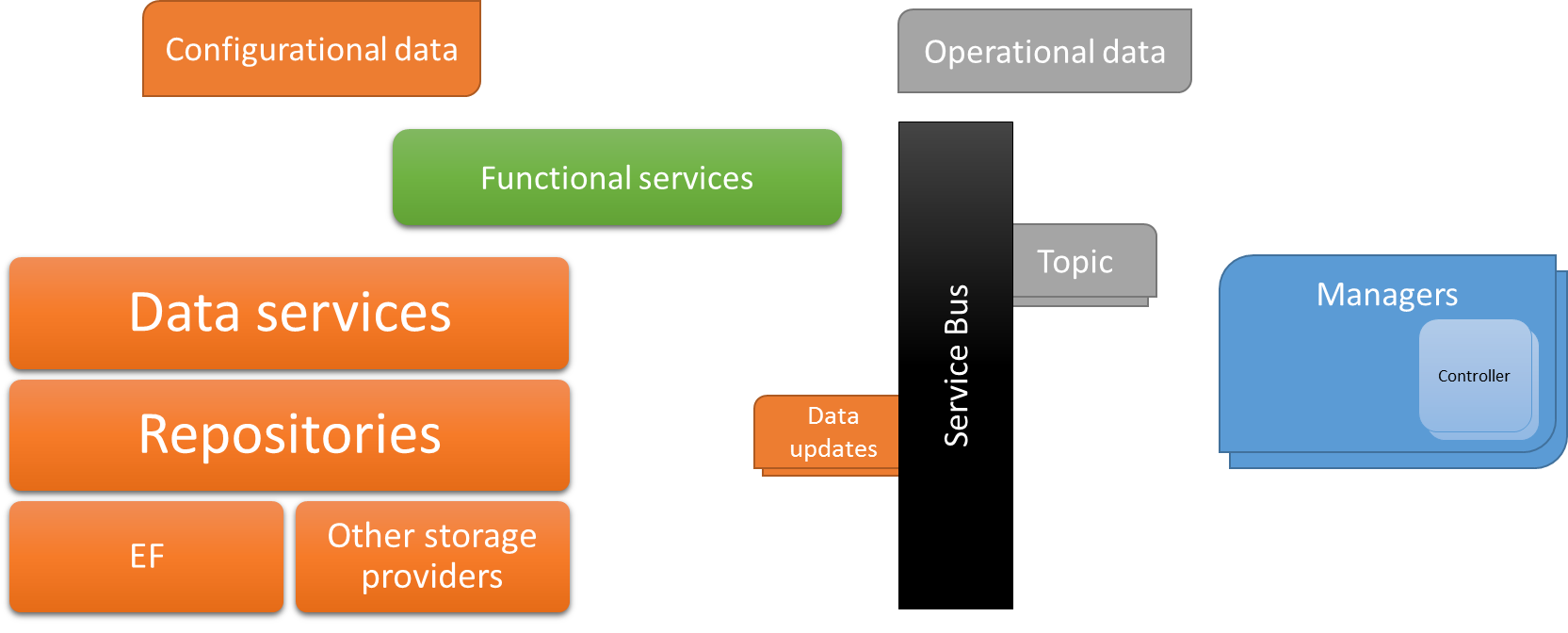
## Issues

During the development of BackgroundSystem 1.x the following issues raised up:

|  |  |
| --- | --- |
| Issue | Description |
| “Monolithic” interfaces | Services contain many entities related to the same area (for instance MembershipService has operations for tenants, users, etc.) |
| Mixed data/functional operations | “Monolithic” services , included CRUD operations and actions/functional operations |
| No updates for specific properties | It’s not possible to request/receive updates for specific properties (for instance, |
| Missing producer/consumer queue scenario | Some producer/consumer scenarios are handled internally, but the concept is totally missing in a distributed environment |
| Embedded managers | All managers are embedded in the IIS application |

# BackgroundSystem 2.0

## Overview



Errata Corrige: The “ServiceBus” block was renamed to “Notifications” block

## Configurational vs Operational data

Configurational data represent the configuration of entities, completed with some “status” information. A valid example is the Unit entity, which contains many properties rarely changing (name, network address) and some properties often changing (for instance, online status). These entities should be handled through the Updatable data services.

Operational data contain volatile information typically involved in a producer-consumer scenario. A valid example is given by Alarms, either generated by the unit or by the system itself.  
These data will be handled though service bus.

## Notifications

We will identify as “notifications” all data transferred through the notification managers, implemented with Medi.

The Medi implementation is appropriate when low latency is required, making it appropriate for client-side applications.  
When transferring these data all references should be represented only by their Id (if available) or, alternatively, by a unique key.

For instance, an Alarm notification could contain the following properties:

* Guid AlarmId
* int Unit\_Id
* string Description
* …

In this example, there is no Unit object available (as it is the typical case when working with DTOs), but only its identifier.  
If required, relational integrity must be anyway ensured by the creator of the notification (valid Unit\_Id).

Please also remark that identifiers for notifications are Guids because they can be generated by the sender. When added to the database, it is still possible to assign the progressive numeric (integer) identifier. To remark the distinction, we’ll have Guid identifiers in the form of <Message>Id (instead of just Id). In our case, the Guid is the AlarmId property.

## What’s included in version 2.0

Only data services will be completely implemented in version 2.0, including their automating update/refresh system.

For the functional services, only Membership and the new Resource service (required by the Update system) will be implemented.

## Data services

Data services offer CRUD interface to store and retrieve data. Operations:

* Add
* Delete
* Update
* Get
* Query(Filter)

These services will be entirely designed with automatically generated code.

## Extended data services

These services extend the generated data services wherever specific needs require the addition of ad-hoc operations (for instance, heavy queries).

Extensions must be carefully evaluated before being added to keep the interface clean.

## Updatable data services

These services are wrapper against normal data services, returning updatable entities implementing an interface similar to INotifyPropertyChanged. Specific clients will be able to get notifications with changes for properties.

## Functional services

Functional services are used to expose any kind of operation which accomplish a certain functionality (for instance: Start unit).

## BackgroundSystem 2.x

Implementation of functional services is not part of BackgroundSystem 2.0.  
Anyway, a rough overview of those services was helpful to design foundations (data services) that are included with this release.

In the following table you can find an overview of the migration for the functional services[[1]](#footnote-1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1.x | 2.x | Action | Description | Effort | Remarks |
| AlarmService | Alarm daemon | Migrated | Alarms created as notifications and then stored using data services | Low |  |
| AlertService | Alert daemon | Migrated | Alerts created as notifications | Very low |  |
| DomainService | - | Removed |  | - | Using (extended) data services to store/load entities |
| Itcs |  | Migrated | Introducing Notifications and | High | Notifications for “DataReady”, Chaching for data |
| Maintenance | - | Removed |  | - | Part of “Admin” |
| Membership | Membership | Migrated |  | Low |  |
| Operation |  | Migrated |  | High |  |
| Unit |  | Migrated |  | Medium |  |

1. Requirements and design subject to changes [↑](#footnote-ref-1)