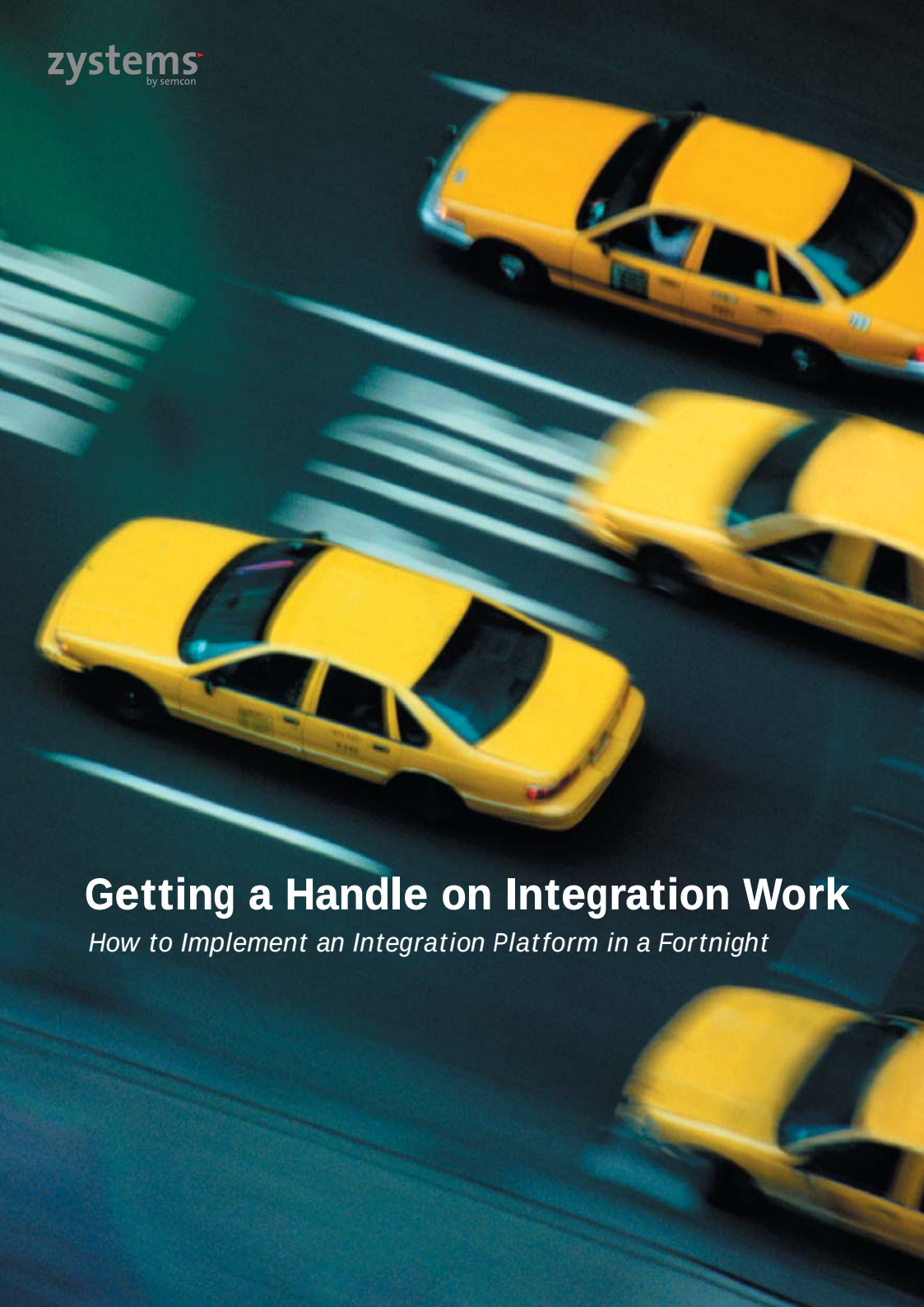
zystems
by semcon



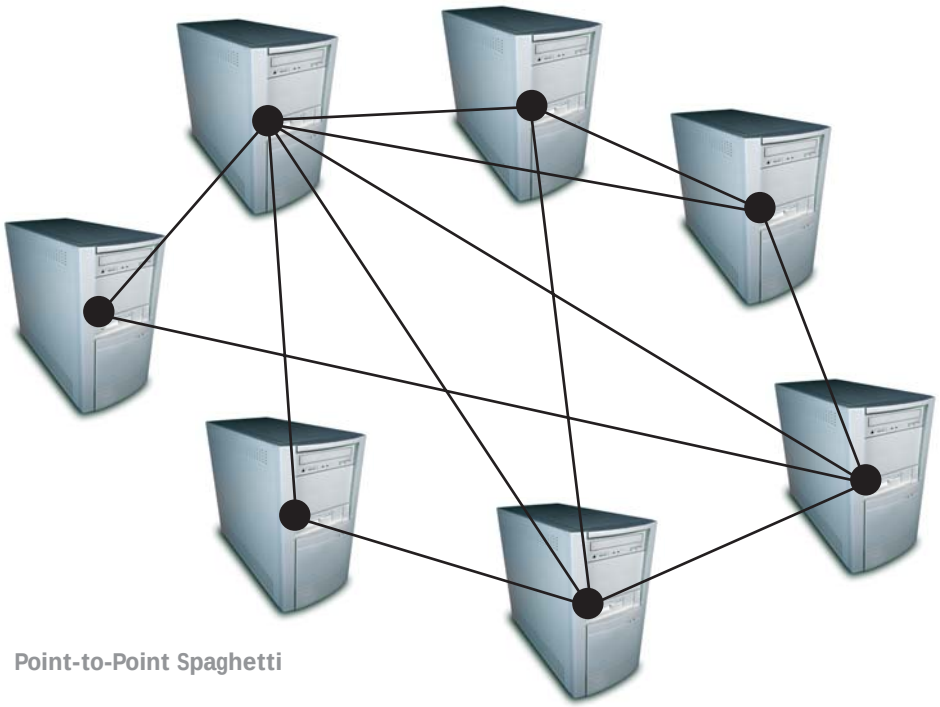
Getting a Handle on Integration Work

How to Implement an Integration Platform in a Fortnight

Contents

Introduction Integrating IT systems has traditionally been a complex and arduous task, and will remain so for the foreseeable future. Using sound methodology when tackling systems integration can substantially alleviate the complexity and avoid some of the pitfalls involved. This white paper outlines some of the key features of such a methodology.

It also describes the Baseline methodology that enables an organization to get started with a fully implemented integration platform in 2–3 weeks.



Point-to-Point Spaghetti

The Problem

If you've ever tried to get more than a few IT systems to communicate with each other, you're familiar with all the problems involved. Over time, you end up with an almost organic web of interdependencies. Obscure, poorly documented point-to-point connections (or interfaces), involving a bewildering collection of formats, protocols, and technologies, make up an infrastructure that only the most courageous dare tamper with.

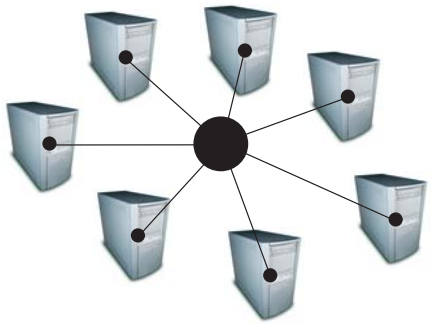
One of the reasons for this is that most of the integration work has been done as part of the various projects dealing with developing and implementing the systems being integrated. This way, no coherent view on system integration exists. Each interface will be designed and implemented from scratch and choice of methods and technology will be entirely up to the people involved in the different projects.

As more and more demand is placed on the IT infrastructure to be flexible and responsive to new business demands (agile is a popular buzzword for this), the spaghetti approach becomes less and less satisfying.

Towards a Solution

Some years back, a way of dealing with this problem emerged in the shape of an architectural mode called Hub and Spoke. This architecture was built around the notion of having one central piece of software (the Hub) deal with all the complexities of inter-system communication. A new class of software, called Message Brokers, entered the stage, providing the technology to build those hubs.

Deploying this kind of middleware is, in itself, not a guaranteed way to integration nirvana. Although you now have a bunch of software components that can help you connect your systems with each other, nothing stops you from keeping the point-to-point mindset, and you may end up creating a somewhat more coherent and smaller bowl of spaghetti. It will still lack the consistency that is required if you want control and transparency.



Hub and Spoke

The Enterprise Service Bus (ESB)

These days, we refer to the hub as the ESB, or Enterprise Service Bus, and we clearly define its responsibilities:

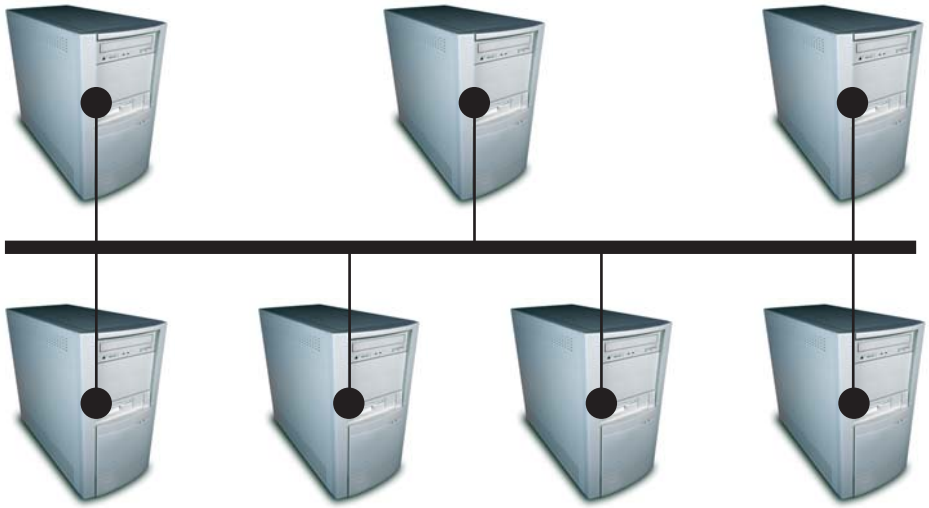
1. Connectivity. The ESB should be able to transport data using one of a clearly defined small set of protocols, such as HTTP or some message queuing protocol (e.g. JMS).

2. Protocol Conversion. The ESB should be able to convert data sent over one protocol, so it can be forwarded over another.

3. Data Transformation. Different systems will send and receive data in different formats, such as fixed length flat files or XML. The ESB should be able to transform (or map) data freely between such formats.

4. Routing. A key feature of any sound integration architecture is loose coupling, meaning that the sending application should know as little as possible about the receiving application. Therefore, no data is sent directly between sender and receiver, but is always routed by the ESB.

These activities are collectively called Message Mediation. As simple as it may seem, designing and implementing an ESB around these four key capabilities is complex and requires careful design. Without a solid methodology, the full potential of the ESB will not be realized.

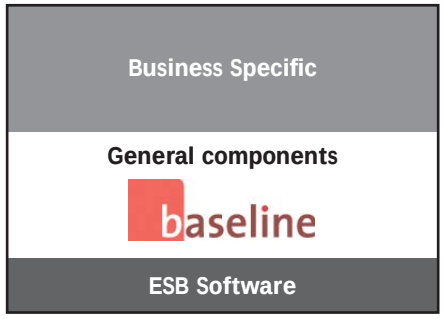


Enterprise Service Bus

Methodology Key Features

After having completed a number of integration projects, you come to the not-so-startling conclusion that a lot of the stuff you do is done in pretty much the same way regardless of the systems involved. You map from one format to the other. You connect the ESB to various systems by reading and writing files in directories, or reading and writing records to database tables. You document your work. You deploy similar artifacts in similar ways to the same environments (e.g. test environment and production environment). A sound methodology should capture those general components and make them available to all people involved in integration projects. Examples of such components include reusable software, document templates, procedures, and roles.

If you think you've seen this before, you may well be right. RUP (Rational Unified Process) is an example of such a methodology. RUP, however, is an all-purpose software engineering discipline, not immediately applicable to systems integration work (or any other specific software engineering endeavor, for that matter). What we will focus on in this white paper are the specific features of a systems integration development methodology.

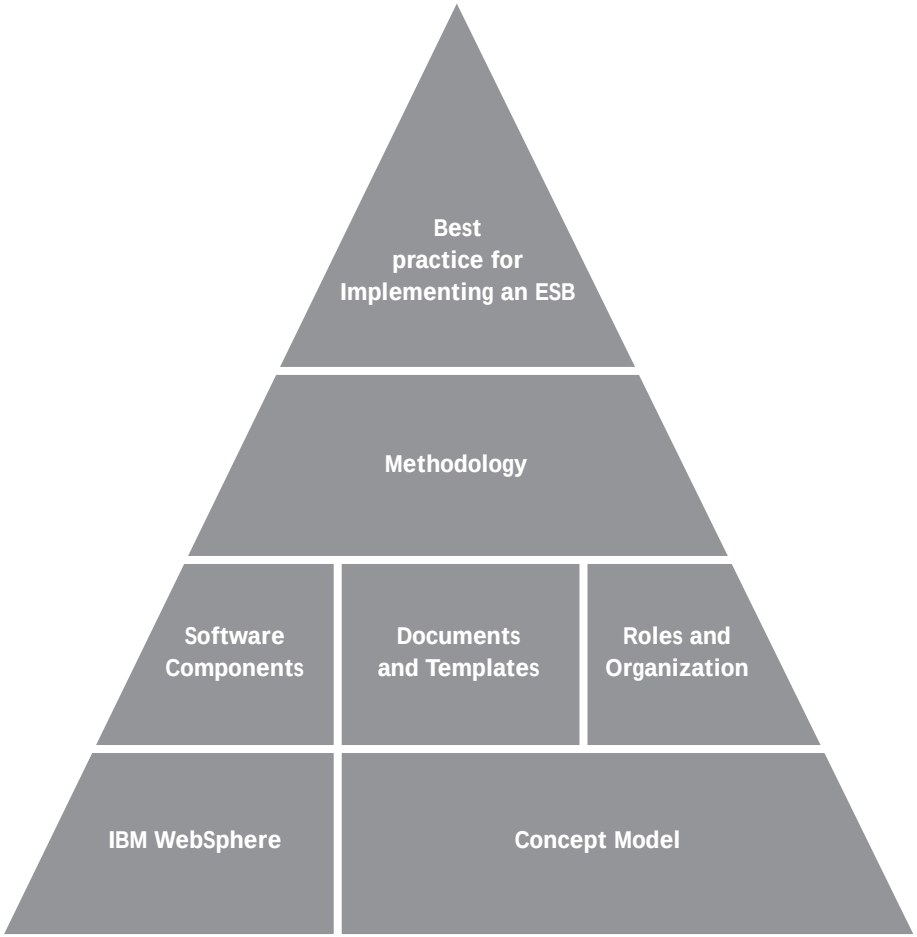


An Integration Project

Baseline

Baseline (developed and marketed by Zystems by Semcon) is a methodology that distills tens of thousands of hours spent integrating IT systems of some of the largest enterprises in Scandinavia into a lean packaged Best Practice for implementing an ESB. The picture below illustrates the components of the Baseline methodology.

Baseline is a Best Practice for implementing an ESB



Baseline

Concept Model

A sound ESB will have a coherent design that permeates all solutions deployed on it. In order to achieve this coherency, Baseline defines a small number of key objects and their relationships to each other. Every solution will be modeled using those objects. All document templates are structured using the same objects.

Integration work generates a myriad of artifacts, such as documents, source code files, script files, binaries, and so on. Naming these artifacts in a consistent way is no small feat (as anyone involved in designing naming conventions will testify). Structured around the concept model, Baseline includes a comprehensive set of Naming Conventions, ranging from the naming of documents, all the way through the naming of source code files, and how to structure and name folders in a versioning system.

An example of such an object is the Contract. The Contract explicitly defines the relationship between the Consumer (another object in the Concept Model) of a Service and the Service itself.

The Concept Model provides consistency in designing the ESB

Documents and Templates

Documenting an ESB in a consistent way is crucial for achieving a number of important goals:

- Maintainability. Supervising and managing the ESB is a complex art, since it will involve a number of operating systems, servers, software components, and contacts with system owners. Having a set of consistently designed documents, all kept in the same place, describing all solutions deployed on the ESB is absolutely essential for this task.
- Reusability. As elaborated later in this white paper, a sound ESB is an ideal foundation (actually, it's more or less a prerequisite) for building a SOA (Service Oriented Architecture). One of the key promises of SOA is the concept of reusable services. However, a service poorly documented, and hidden away on some shared disk in a network, is not likely to be identified by new integration projects on the lookout for reusable services. Having all services documented in a consistent manner in a central repository is a minimum requirement for service reuse to happen.
- Quality. Working with design and implementation supported by document templates ensures that all relevant questions are asked at an early stage (providing the creation of the documents is a living part of the design and implementation process rather than an afterthought).

Baseline provides a comprehensive set of templates and documents covering all aspects of an integration project, including:

- Requirements collection
- Project startup
- Solution design and implementation
- Testing
- Packaging and deploying solutions
- Supervising solutions in a production environment
- Governance (i.e. the centralized organizational function of maintaining consistency across integration projects)

Documenting the ESB ensures Maintainability, Reusability, and Quality

Roles and Organization

Defining a number of roles, and delineate their responsibilities, is a crucial part of any successful software development effort. Baseline defines a detailed process for producing deployable solution packages, and identifies a number of roles. Having clear demarcation lines between the roles ensures a concept know as Separation of Duties.

As an example, the project team involved in producing a solution package is never allowed access to either QA (acceptance test) or production environments. This means that the people responsible for producing the code are physically unable to put it into production. Instead, a well defined process of handing the solution package over to the operations staff is defined in Baseline. After verifying the completeness and quality of the package, operations personnel are then responsible for deploying it.

Clearly delineated Roles ensures Separation of Duties

IBM WebSphere and Software Components

Although Baseline is not tied to any specific ESB software vendor, Zystems by Semcon works exclusively with IBM WebSphere business integration software, making us the leading consultancy outfit in this niche in Northern Europe. Comprehensive as it may be, WebSphere business integration software still lacks certain features, often involving small subsets of large, expensive WebSphere packages. Zystems by Semcon also provides a number of software components complementing IBM WebSphere offerings. Examples include various adapters to connect applications to the ESB; a testing tool called Testline, used to build automated test suites for unit and regression tests; and a transaction tracking application called Baseline Track & Trace, enabling complete transactions to be tracked through the ESB and searched using logical identifiers, such as order numbers or article numbers.

Recurring Software demands, such as ESB connectivity, Automated solution testing, and Transaction tracking, are included in Baseline

Baseline Start-kit

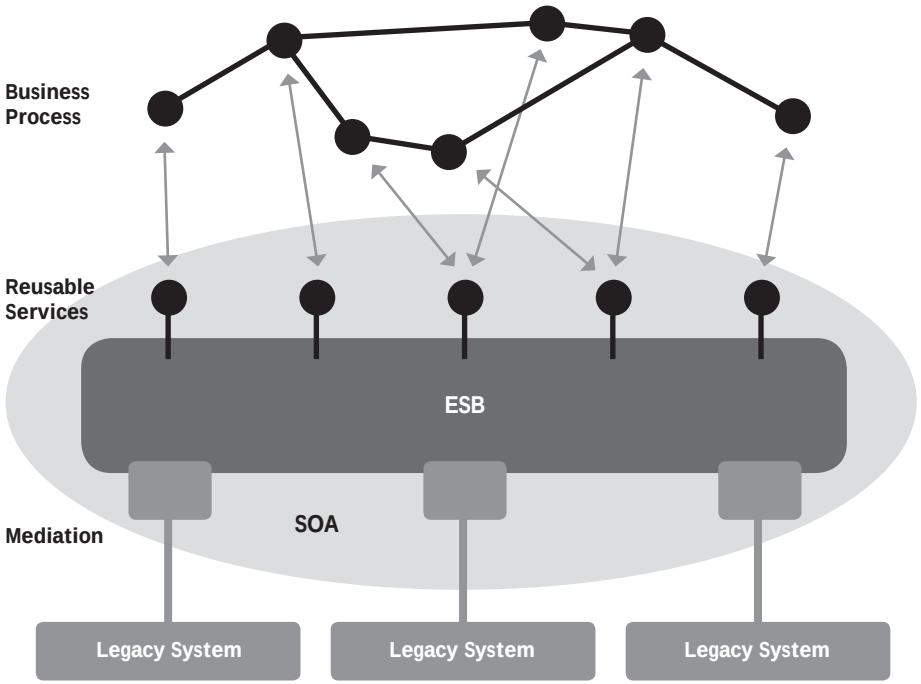
The Baseline Start-kit is a set of services at a fixed price aimed at introducing and implementing an ESB in an organization in 2-3 weeks. It includes:

- Architecturing and installation of relevant IBM WebSphere products
- Training of the customer's staff in using the WebSphere product (as well as Baseline Methodology and Software components)
- Completing a simple first integration project, using the Baseline methodology
- The Baseline deliverables themselves (Document Templates and Software Components)

Using the Baseline Start-kit, numerous organizations have successfully been kick-started into using an ESB.

Baseline Start-kit implements an ESB in 2-3 weeks

Baseline and SOA



Baseline in a SOA perspective

The Baseline Concept Model relies heavily on the Service concept. Although this white paper does not go deeper into the SOA jungle, we will touch briefly on the relationship between Baseline and SOA.

Using Baseline methodology, you are able to build a consistently designed ESB exposing a number of reusable services. The picture above shows how this may look.

Baseline will help you design, implement, and document the entire shaded area, i.e. the complete SOA.

It is important to note that Baseline in no way dictates a full-fledged SOA, but rather enables and supports the SOA paradigm.

Also, note that Baseline currently does not contain any explicit support for the design and implementation of Business Process Automation (BPA). In enabling the construction of a sound ESB, Baseline is rather the foundation (from a technological point of view) for BPA.

Baseline supports SOA and is the foundation for Business Process Automation

Summary

This white paper presented the woes and perils of integrating IT systems, and the Hub and Spoke architecture and the Enterprise Service Bus (ESB) where introduced as possible solutions.

As we saw, designing and implementing an ESB is a complex task that needs careful planning and the support of a sound methodology. Some key features of such a methodology were presented:

- Based on a clear Concept Model
- Including Documents to support all aspects of integration work
- Clear definitions of Processes and Roles

Baseline was introduced as a methodology delivering all those key features, and more.

Baseline Start-kit was presented as a set of services combined with the methodology to kick-start an organization into the successful implementation of an ESB.

Rounding up, Baseline was presented as an aid in building a SOA (Service Oriented Architecture) as a foundation for Business Process Automation.

For more information

If you want to learn more about Baseline and Zystems by Semcon, visit our web-site at www.zystems.se.

zystems
by semcon

Theres Svenssons gata 15, 417 80 Göteborg, Sweden. +46 (0)31- 721 00 00
Kaprifolievägen 1, 227 38 Lund, Sweden. +46 (0)46-270 83 00
Patentgatan 8, 112 67 Stockholm, Sweden. +46 (0)8-562 906 00
www.zystems.se