Sitecore 7.5

xDB™ Overview and Architecture

A conceptual overview of the architectural changes introduced in Sitecore 7.5

Table of Contents

Chapter 1 Introduction ... 3
Chapter 2 Experience Database (xDB) Architecture ... 4
2.1 Overview ... 5
2.1.1 Key Components ... 6
2.1.2 Data Flow ... 7
2.2 Scalability Options ... 8
2.2.1 Minimal Environment ... 8
2.2.2 Vertical Scaling ... 8
2.2.3 Horizontal Scaling ... 10
2.3 Collection Database (MongoDB) ... 13
2.3.1 MongoDB and NoSQL ... 13
2.3.2 MongoDB Example Architecture ... 13
Replication ... 13
Setting Priorities ... 14
Example Architecture ... 15
Sharding ... 15
2.4 Session State ... 16
2.4.1 Overview ... 16
2.4.2 Private or ASP.NET Session State ... 16
2.4.3 Shared Session State ... 17
2.5 Processing and Aggregation ... 18
2.5.1 Types of Processing ... 18
2.5.2 Processing Overview ... 18
2.5.3 Continuous Update of the Reporting Database ... 19
2.5.4 Rebuilding the Reporting Database ... 19
2.6 Reporting ... 21
2.6.1 Reporting Architecture ... 21
2.6.2 Report Queries ... 22
Chapter 3 Glossary of Terms ... 24
3.1 Glossary of Terms ... 25
3.1.1 Sitecore Experience Platform ... 25
3.1.2 Experience Database ... 25
3.1.3 Experience Profile ... 25
3.1.4 Experience Marketing ... 25
3.1.5 Contact ... 25
3.1.6 Device ... 26
3.1.7 Interaction ... 26
3.1.8 Online Visit ... 26
3.1.9 Account ... 26
3.1.10 Collection Database ... 26
3.1.11 Reporting Database ... 27
3.1.12 Fact Tables ... 27
3.1.13 Dimension Tables ... 27
3.1.14 Aggregation ... 27
3.1.15 Rebuild of the Reporting Database ... 27
3.1.16 Reporting Service ... 28
3.1.17 Session State Server ... 28
3.1.18 Private Session State ... 28
3.1.19 Shared Session State ... 28

Chapter 1

Introduction

The Sitecore Experience Database (xDB™) collects all online and offline customer interactions from all channel sources in a real-time big data repository. It connects interaction data to create a comprehensive, unified view of each individual customer, and makes the data available to marketers to manage the customer experience in real time.

The Sitecore® Experience Platform™ is Sitecore CMS and xDB combined in one, powerful solution, from the content management capabilities of the CMS to the experience marketing, scalability, and performance features of the xDB.

The xDB is a flexible and scalable architecture that is capable of handling very large amounts of customer data. You can scale both up and down from small single server, low traffic solutions to larger solutions consisting of hundreds of servers designed to handle high volumes of traffic.

This document contains the following chapters:

- Introduction
- Experience Database (xDB) Architecture
- Glossary of Terms

Chapter 2

Experience Database (xDB) Architecture

This chapter gives a high-level conceptual overview of the xDB and provides some guidance on installing a typical Sitecore xDB architecture.

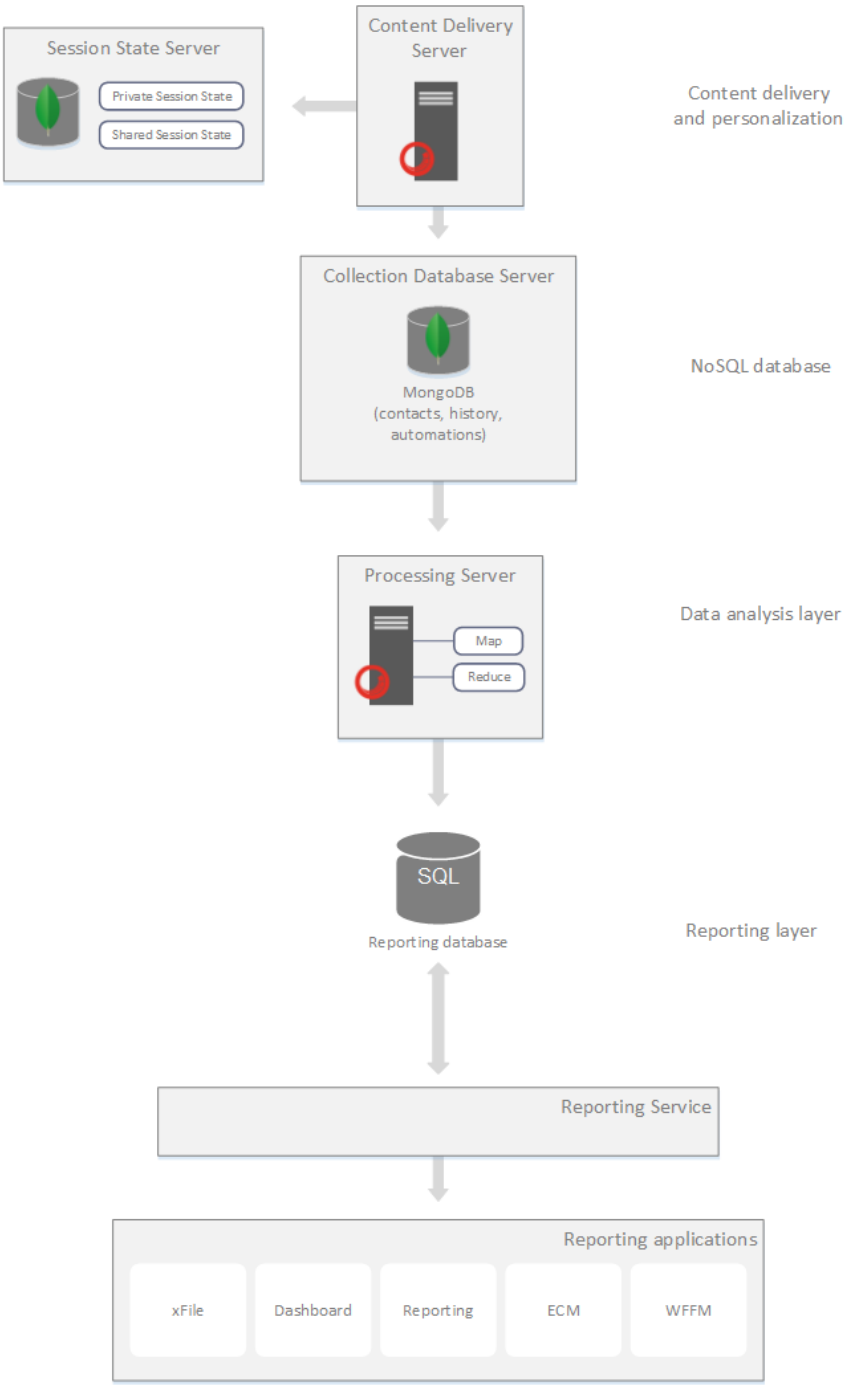
This chapter includes the following sections:

- Overview
- Scalability Options
- Collection Database (MongoDB)
- Session State
- Processing
- Reporting

2.1 Overview

When you install Sitecore xDB for the first time, use the information in this section to become familiar with all the key components.

The xDB is a flexible architecture that provides you with all the building blocks you need to deploy single-server low-traffic solutions to highly scalable high-traffic solutions.



2.1.1 Key Components

Key components in the Sitecore Experience Platform:

- Content delivery (including personalization).
- Content management.
- CMS databases (SQL Server).
- Session state server.
- Collection database (MongoDB).
- Processing server.
- Reporting database.
- Reporting service.

Content Delivery Server and Personalization

Content delivery and personalization is implemented using a Sitecore application server that serves incoming HTTP requests from the Internet.

Content Management Server

Content management in this document refers to the core functionality of Sitecore such as authoring and editing content, managing content and configuring your website solution.

CMS Databases (SQL Server)

The following three Sitecore databases are implemented in Microsoft SQL Server:

- *Core* – contains all settings, like a large configuration file, for the Sitecore user interface. You can use it if you are customizing Sitecore, such as adding new applications to the Sitecore desktop.
- *Master* – contains all versions of all content. This is where business users author and edit content.
- *Web* – contains the latest version of published content that has reached a final workflow state.

Session State Server

The session state component is a session state store used by content delivery and personalization. This is a standard ASP.NET session state store provider that includes support for the `Session_End` event. In Sitecore xDB, the default session state provider is `inProc` ASP.NET but it also comes with two additional providers, one for MongoDB, and another for SQL Server.

Collection Database (MongoDB)

The collection database (MongoDB) is the primary storage of all collected analytics information as well as being the registry of contacts and engagement automation states. It is implemented using MongoDB, a highly scalable document-based NoSQL database.

Processing Server

The processing and aggregation component extracts information from captured, raw analytics data and transforms it into a form suitable for use in reporting. It also performs specific tasks on the collection database that involve mass updates.

Processing and aggregation is implemented using a Sitecore application server connected to both the collection and reporting databases. It can also run on the same Sitecore server as other components. You can configure processing or aggregation on single or multiple servers to achieve higher performance on high-traffic solutions.

Reporting Database

The reporting database stores aggregated information from the collection database suitable for fast querying and reporting. It is implemented in Microsoft SQL Server.

Reporting Service

The Reporting Service API allows you to execute queries and extract information from the collection and reporting databases. It is a part of the xDB but you can also configure the Reporting Service as an intermediate reporting server which performs processing tasks and offloads endpoint application servers.

Note
The database names *collection* and *reporting* are not official Sitecore product names.

The collection database refers to the functionality and purpose of the MongoDB `analytics` database.

The reporting and reporting.secondary databases correspond to the Microsoft SQL Server `Sitecore_Analytics` and the `Sitecore_Analytics_Secondary` databases. These are legacy names that have not been changed for this release.

2.1.2 Data Flow

How data flows through the xDB:

1. A contact decides to interact with a website.

   For more information about devices and types of interactions, see the definitions for *Device* and *Interaction* in the Glossary of Terms section.

2. Depending on geographic location of the contact their visit is redirected to the closest cluster (datacenter) through DNS configuration.
3. Load balancing software directs the contact to an appropriate Sitecore content delivery server. The contact stays connected to the same cluster even if they switch devices or use another browser. A contact can only move to another cluster if all their active sessions expire and all information is then saved to the collection database.
4. During the session, details of the contact, the interaction, and the device are stored in either private or shared session state. On *SessionEnd*, this data is flushed to the collection database and scheduled for processing.
5. Customer data in the collection database is aggregated by the processing layer. The processing layer reduces and groups interaction (online visit) data and stores it in the reporting database for use by the reporting layer. Data is also aggregated by the processing layer for use by the segmentation index.
6. The Reporting Service queries the reporting database and the collection database to fetch reporting data for use in applications such as *Engagement Analytics* and the *Executive Insight Dashboard*.

2.2 Scalability Options

There are several ways you can install and configure Sitecore environments to scale vertically and horizontally to cope with increased demand and high amounts of traffic on your website.

The xDB is flexible enough to run as a standalone environment or can be scaled vertically or horizontally. This section does not recommend a single approach but outlines three different possibilities which you can implement together to achieve better performance and scalability.

2.2.1 Minimal Environment

A minimal installation is a standalone, all-in one configuration when all components are installed on the same computer.

This setup is most suitable for development and testing. It is recommended that you do not use a minimal installation as a production environment.

For example, a minimal installation could include the following components:

- An application server with a Sitecore instance that has all the xDB components enabled for content management, content delivery, processing, reporting, and session tracking.
- Collection database (MongoDB) – to record the entire customer experience.
- Reporting database (SQL Server) – to provide reporting data for the *Executive Insight Dashboard* and *Engagement Analytics* reports.
- Content databases (master, core, and web).

2.2.2 Vertical Scaling

Definition: "*To scale vertically (or scale up) means to add resources to a single node in a system, typically involving the addition of CPUs or memory to a single computer*". (*Wikipedia*)

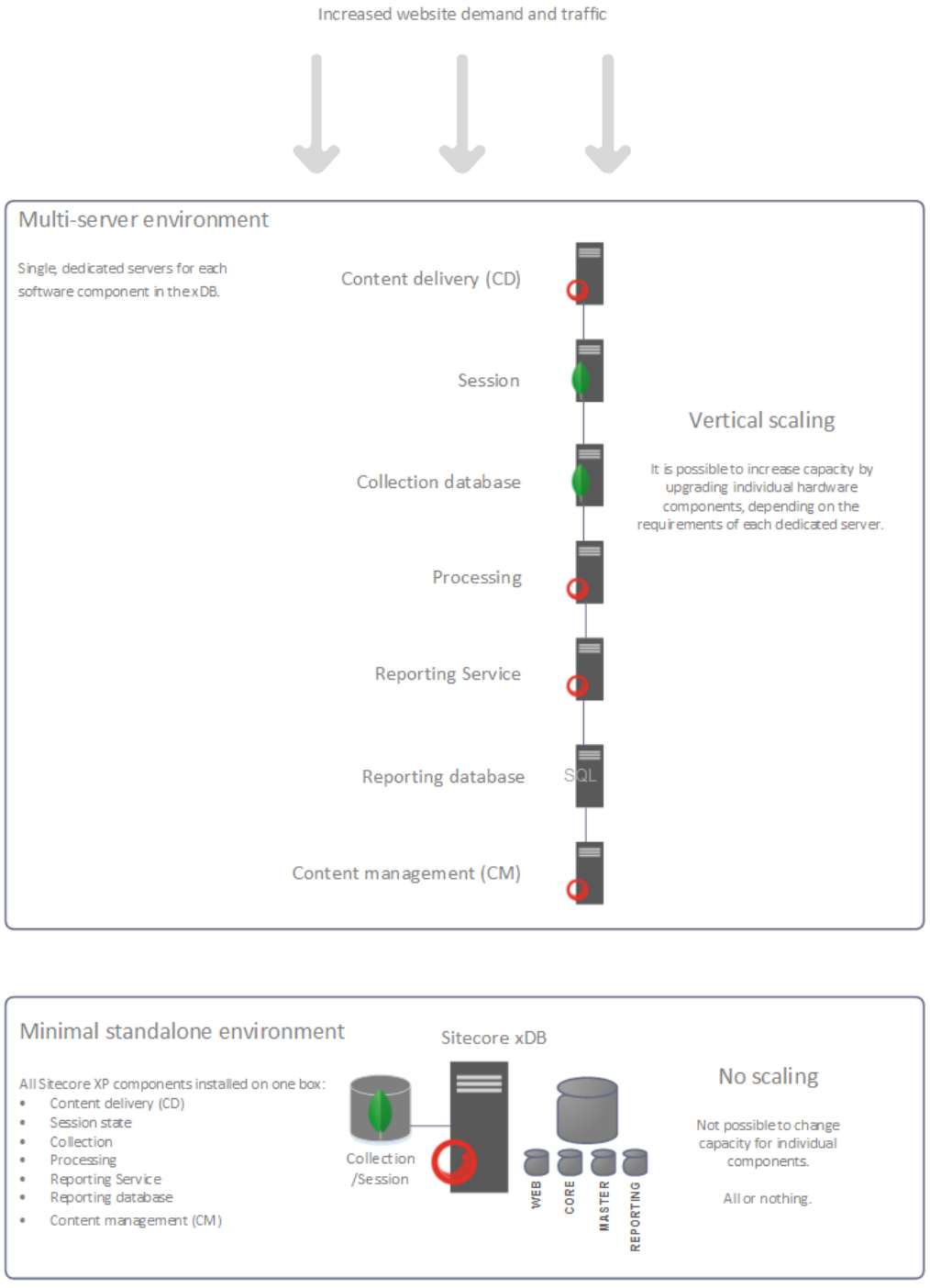
In Sitecore xDB, you could start this process by installing single instances of each component on separate servers, therefore moving from a single server to a multi-server environment.

For example, you could start by having three separate servers for:

- Content delivery
- Collection
- Reporting

An advantage of vertical scaling is that you can upgrade specific components that need an upgrade (depending on the task assigned to the server) without upgrading the entire environment. For example, a content delivery server might need faster CPUs while the collection database needs more disk space for storage and plenty of RAM to maintain system performance. Therefore if you scale vertically, you can dedicate specific servers to specific tasks and upgrade hardware accordingly.

Sitecore xDB vertical scaling overview.



Vertical scaling is suitable for the needs of most organizations, particularly small to medium sized organizations where the data requirements are not too high.

For example, you could move from a single server to multi-server environment by:

- Deploying collection, reporting and content databases on separate servers.
- Deploying application components, such as content delivery, content management, and processing on separate servers.

2.2.3 Horizontal Scaling

Definition: "*To scale horizontally (or scale out) means to add more nodes to a system, such as adding a new computer to a distributed software application".* (*Wikipedia*)

In Sitecore, this means you deploy multiple servers for specific components such as content delivery, processing, content management, or collection to increase the capacity of your solution.

For example, as demand increases, you could deploy multiple clusters of content delivery servers to deliver web content to contacts as quickly and efficiently as possible. On multiple content deliver clusters you can use load-balancing software to optimize performance across clusters and to maintain high availability. You could also implement multiple content delivery clusters to provide multiple geographically distributed collection points.

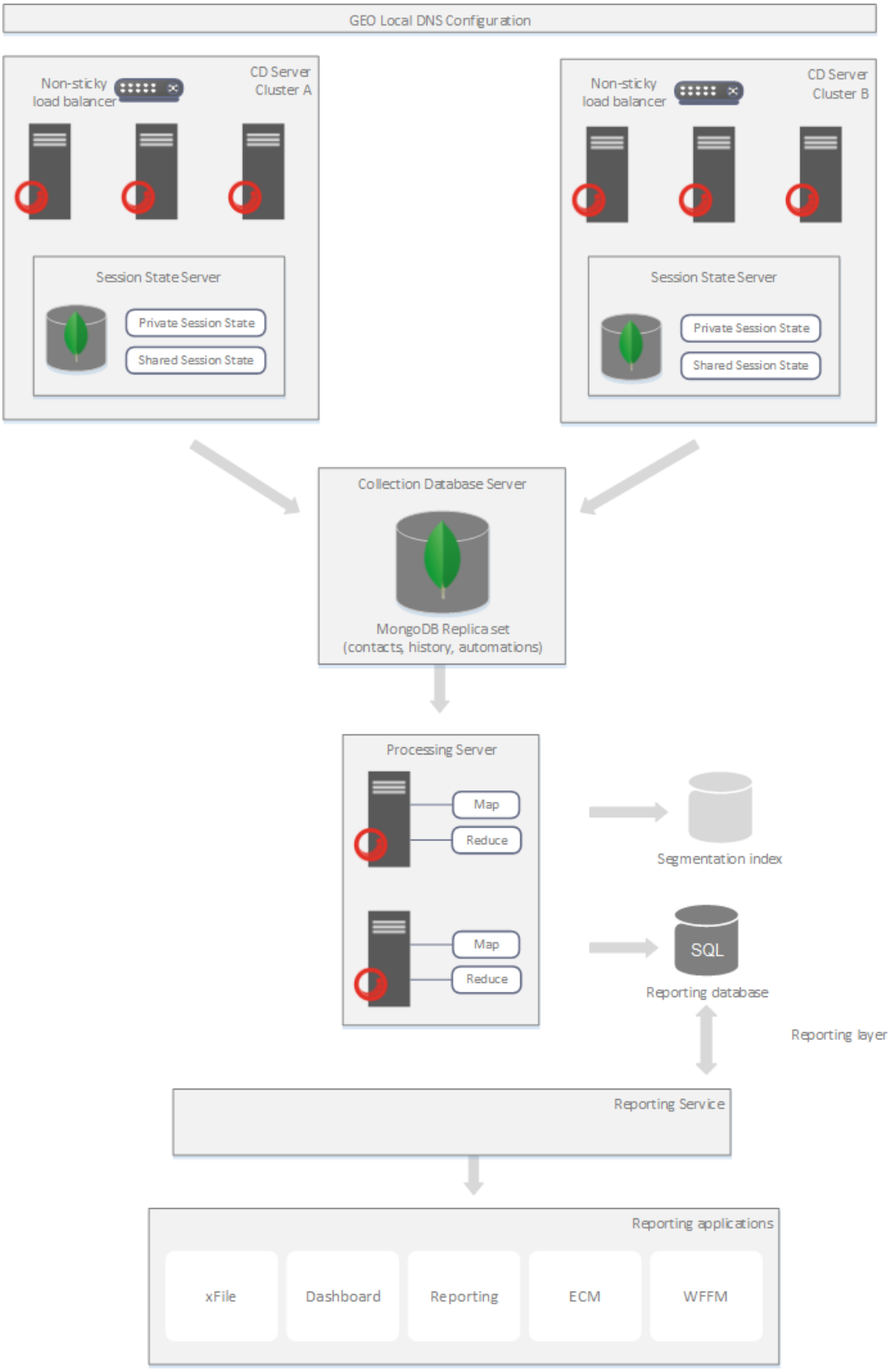
Sitecore architecture is designed for scalability and high performance regardless of the size of your website or the organization that it serves.

Horizontal scaling is particularly suitable for large to enterprise organizations that need to handle a lot of website traffic and want reliability and high availability. These types of organizations often have large data storage requirements so may need to consider horizontal scaling as part of their future strategy for expansion.

For example, to scale horizontally you can deploy the following:

- Multiple content delivery servers.
- Content delivery servers can be arranged into multiple clusters, either to increase capacity beyond a single cluster, or to create geographically distributed content delivery and traffic collection points.
- A session state server for handling session data. This is required if you have a cluster with more than one node.
- Collection database (MongoDB) for storing contact and interaction data– this can be a single MongoDB replica set (which requires a minimum of three nodes, for example two data nodes and an arbiter) or it can be a sharded cluster with multiple replica sets.
- One or more processing servers. The number of processing servers you have depends on how much website traffic you need to handle.
- Multiple content management servers –- depending on your content editing needs.
- Reporting server that runs the Reporting service hosted on a separate Sitecore server instance.
- Reporting database hosted on a separate Microsoft SQL Server instance (this could also be a failover cluster).

Scalable Sitecore xDB architecture.



Benefits of deploying a fully scalable xDB architecture:

- High flexibility – from single server to highly scalable solutions that can expand and grow, by scaling up and scaling out as the demand on your website increases.
- High performance – For example, you can optimize the performance of multiple content delivery servers using load balancing software and have the possibility to scale almost any component vertically and horizontally.
- Full support for content delivery web clusters - You can configure content delivery web clusters and session state out of the box without the need for any extra custom components. You can distribute the load on your servers using load balancing software with failover, and can upgrade your solution while it is still live.
- Highly scalable data storage – A MongoDB database at the core of data collection – provides storage that incrementally scales to terabytes or even petabytes of data. Capacity can be increased using horizontal scaling which in MongoDB is implemented by sharding. You can add more servers over time when you need to expand your solution. Both replication and sharding can be performed across multiple geographical locations. All customer interaction data (online visits) is stored in a MongoDB NoSQL database (nothing is deleted) and is made available to Sitecore reporting applications. Also the processing server can be scaled out to perform all activities necessary for processing and aggregation.
- High availability – all application components can be deployed as multiple servers, and database components support high availability and data persistence. In MongoDB replication enables higher availability (HA) and better fault tolerance.
- Cloud-ready – application components can be deployed as preconfigured servers allowing for quick scaling up and down on demand, depending on traffic patterns.

The advantages mentioned here make the xDB particularly suitable for medium and large-scale enterprises.

For more xDB architectural options and recommended best practices, see the scalability documentation on the Sitecore Developers Network (SDN).

2.3 Collection Database (MongoDB)

The collection database (MongoDB) is a highly scalable database capable of collecting and storing vast amounts of customer experience data. In the xDB, the collection database is used as the central repository for storing contact, interaction, history, and automation data.

In this section we explain how the collection database can help you to increase the availability, scalability, and performance of your Sitecore implementation to handle billions of visits.

To achieve these objectives a scalable NoSQL database solution is required. MongoDB is an open-source, NoSQL, document-oriented database designed for ease of development and scaling.

The main benefits of using MongoDB with the xDB are to:

- Handle billions of visits or interactions per year
- Provide scalability options to increase storage capacity
- Maintain high performance

The collection database (MongoDB) collects and processes analytics data gathered from all your websites. This data can include interactions, contacts, devices, location, automation data, and events triggered such as goals converted or campaigns activated.

2.3.1 MongoDB and NoSQL

MongoDB and other document-oriented database systems require you to think about databases in a slightly different way. Previously in Sitecore all visitor analytics data was collected and stored in Microsoft SQL Server. This approach works well but over time, extracting relevant data for reporting purposes becomes more and more time consuming and expensive, particularly when there are billions of visitors.

In SQL Server, it is only possible to scale vertically (increasing the size of the server) and not horizontally (increasing the number of servers). So to achieve the required improvements in scalability, performance, and availability a solution such as MongoDB is the appropriate choice.

Before you implement Sitecore with MongoDB as your collection Database, see the *xDB Configuration Guide* on SDN for more detailed information and recommendations on configuring MongoDB.

2.3.2 MongoDB Example Architecture

The example configuration in this section includes the minimum possible number of MongoDB instances in a single replica set while still ensuring that replication is in place.

The diagram included in this section outlines a standard MongoDB configuration if you choose to use MongoDB as your collection database.

This example configuration consists of three Azure virtual servers, one primary, and two secondary.

Replication

All production deployments should use replication to provide redundancy and increase data availability. Replication copies data to multiple servers, so if one server fails no data is lost. Servers can be placed in different geographical regions, defaulting to one and using the other as backup for *Disaster Recovery* (DR) or *High Availability* (HA) failover.

Setting Priorities

Depending on how you configure MongoDB, you can increase read capacity by setting priority on each instance. The primary is the only member of a replica set that can receive write operations. You set priority by allocating a number to each instance, the highest number gets top priority.

In the example outlined in this section, the member with priority 10 will always be elected the primary, if it is available. The other member with priority 9 will be secondary, and if you have an arbiter this has no priority setting, as it can never become the primary (it does not have any data and is only used for voting during elections). Similarly, a data member with priority 0 will never become primary, though it still maintains a copy of the data and can serve reads if the appropriate read preferences are used.

If the primary becomes unavailable for any reason, MongoDB performs an election to determine which member becomes the new primary. Only members that have a value higher than zero (0) can be considered in elections and there must be a majority vote.

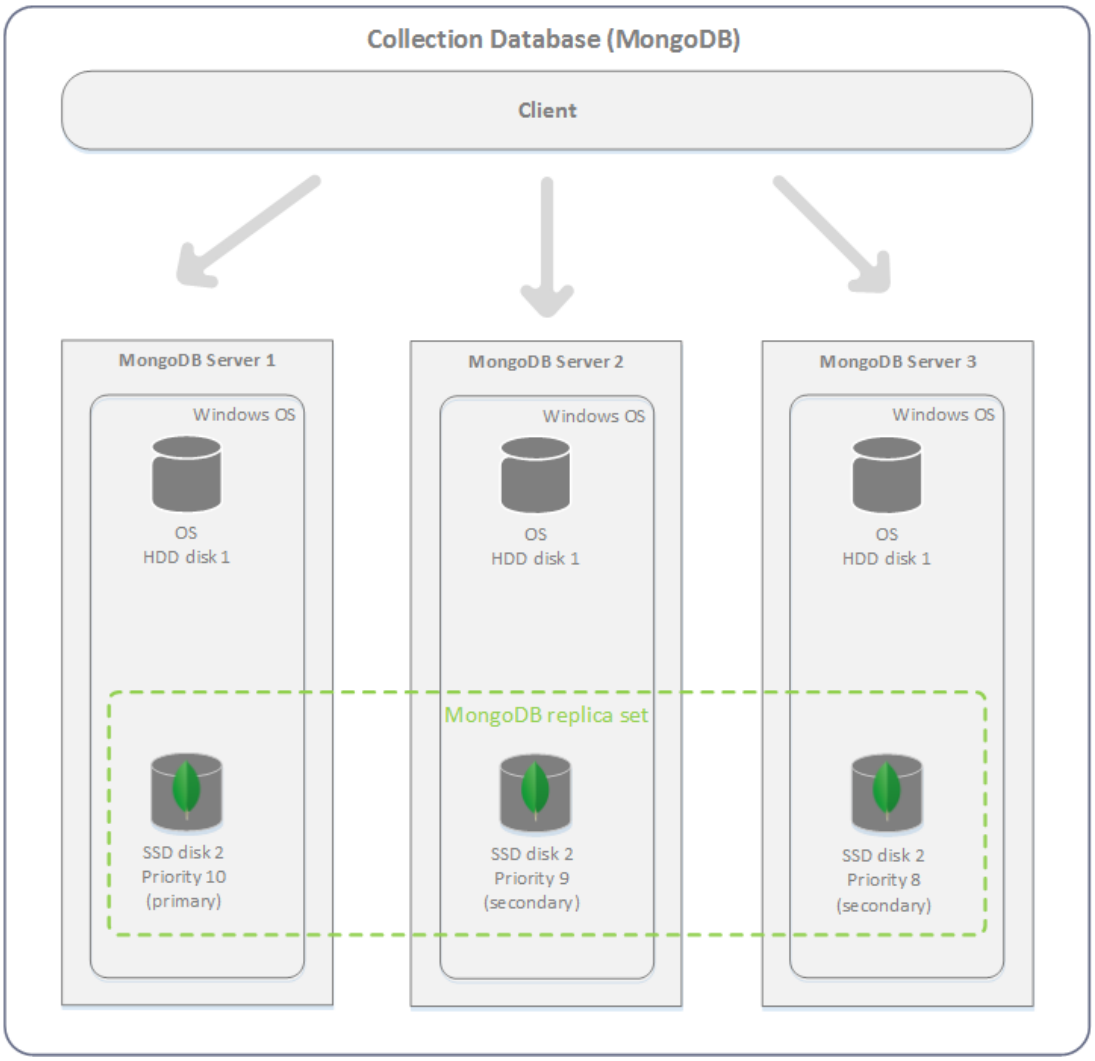
There are many other worthwhile ways in which you can use priorities, for example, to give priority to nodes with the most powerful hardware or to give priority to the primary data center.

Note
We do not recommend setting priorities just to make it more predictable which node will become the primary. This can easily be discovered using the `rs.status()` method and can result in making operations more complicated.

Refer to the MongoDB documentation for more information on replica sets, setting priority, and elections: http://docs.mongodb.org/manual/replication/

Example Architecture

Collection (MongoDB) database – example diagram of a standard MongoDB replica set implementation on three servers (3 data nodes). For reliability and resilience we recommend 3 data nodes (1 primary, 2 secondary), especially if write concern w=majority is used (to avoid rollbacks) and during periods when one of the nodes is unavailable (e.g. during maintenance, upgrading, or defragmenting).



Sharding

As the collection database grows, you will find that you need more storage space. You might also find that read and write speeds slow down. In MongoDB, there are different ways of configuring your environment to cope with these demands. Sharding enables you to scale horizontally. This means adding more computers to store data and spread the load, which can also help to improve read capabilities, write speeds and contributes to generally improved performance.

For more information on installing a MongoDB database with Sitecore, see the xDB Configuration Guide on SDN.

For more information on installation, replication and the different configuration options available in MongoDB, see http://www.mongodb.org/ for MongoDB documentation.

2.4 Session State

In the xDB you can use a session state server to track all contact sessions across different browsers and devices. Configuring session state is particularly important if you have deployed a multi-server, fully scalable environment with clusters of content delivery or processing servers.

2.4.1 Overview

Tracking the session state of your contacts is important regardless of whether you install a standalone environment or a fully scalable environment.

If you decide to deploy a large, fully scalable environment with vertical and horizontal scaling, then session state becomes even more important. For example, in a multi-content delivery cluster environment, you can share contact information across clusters to ensure that contacts stay connected to the same cluster where their interaction originated. This is not possible to achieve without installing a session state server.

In xDB, there are two types of session state:

- Private or ASP.NET
- Shared

2.4.2 Private or ASP.NET Session State

In private or ASP.NET session state, all data related to a specific interaction is collected and saved to the session state server database.

Private session data collected during a session includes:

- Pages visited
- Goals converted
- Events triggered
- Campaigns activated
- Engagement value points accumulated

The default provider is the *in process* (`inProc`) session provider which uses in memory session state. This is suitable for small, standalone installations used for test or development purposes.

If you have a multi-server environment you should use one of the following *out of process* ASP.NET session state providers:

- *Sitecore ASP.NET Session State Store Provider for MongoDB*
- *Sitecore ASP.NET Session State Store Provider for Microsoft SQL Server*

These providers enable you to choose whether you want to use MongoDB or SQL Server session database as your session state store. They also support the `session end` event which is required by the xDB to track website visits reliably.

Note
You should only configure one session state server for each content delivery cluster that you have.

For more information on how to install and configure a MongoDB database for tracking session state see the *xDB Configuration Guide* on SDN or the documentation on the MongoDB website.

2.4.3 Shared Session State

Shared session state is all the data related to a specific contact and devices collected and saved in the session state server database. When you configure shared session state we recommend you use an '*out of process*' provider and choose either the MongoDB session provider or the SQL Server session provider.

Shared session data collected during a session:

- Contact details
- Devices
- Engagement automation states

All data unique to the contact that can be shared across multiple interactions, such as contact details, devices used and engagement automation states triggered. This information needs to remain accessible to other concurrent operations, for example, interactions and background processes and can be shared across multiple sessions.

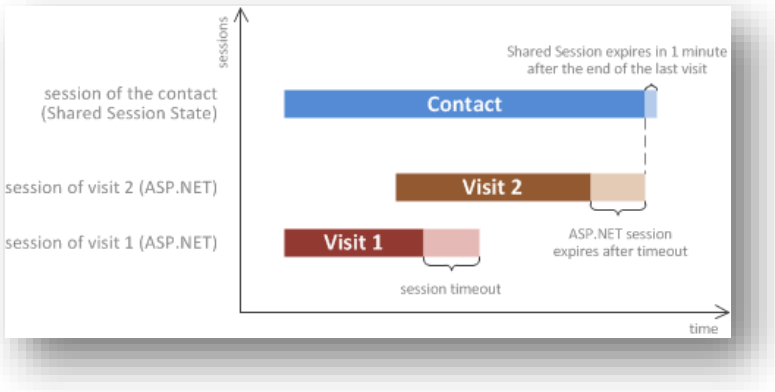
Shared session state is a component that is used by Sitecore for storing information about current contact sessions and related data.

The standard ASP.NET session works in the context of a session cookie stored in the browser. This means that if a contact uses Sitecore from two different devices, or even two different browsers on the same device, then two separate sessions are created.

As opposed to the ASP.NET session state that is exclusive to an interaction, shared session state stores a contacts session and is accessible from any visit session that is associated with the contact.

Session Lifetime (Shared)

Graph showing the lifetime of the sessions for a single contact and two concurrent interactions made from two different devices.



Note
When there is only a single standalone content delivery server, shared session state can be stored in memory because there is no need to share the session state with other servers. In this scenario, shared session state is still useful as a cache of contact states, helping to reduce traffic between the application and the database.

For more information on configuring shared session state, see the *xDB Configuration Guide* on SDN.

2.5 Processing and Aggregation

Sitecore xDB uses two types of databases:

- A MongoDB (NoSQL) collection database which collects all captured experience information in a loosely structured way.
- A SQL Server reporting database which stores information extracted from the collection database in a form suitable for reporting.

By default Sitecore xDB must always keep the reporting database in sync with the collection database.

2.5.1 Types of Processing

Sitecore xDB contains a data analysis layer that processes data from the collection database. The data analysis layer can perform several different kinds of processing on data stored in the collection database:

- Aggregation – data from the collection database is extracted, grouped and reduced and then stored in the reporting database for use by Sitecore reporting applications.
- Continuous update – this process always keeps the reporting database in sync with the collection database.
- Rebuilding the reporting database – this process rebuilds the entire reporting database upon request.
- Maintenance - performs routine maintenance tasks on the collection database.

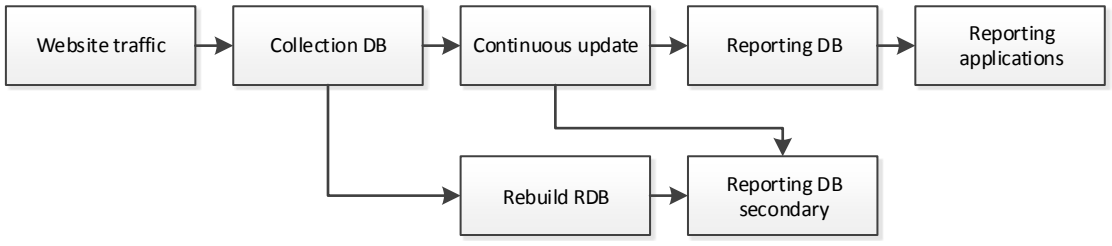
2.5.2 Processing Overview

In daily operation the xDB uses a single reporting database that is continuously synchronized with new traffic information that is added to the collection database.

Continuous Update of the reporting database

This diagram shows the processing required to keep the reporting database continuously synchronized with the collection database.

Rebuild of the reporting database



This diagram shows the processing required to rebuild the entire reporting database on request.

In order to minimize interruptions to reporting functionality, the rebuild process works with a second instance of the reporting database called reporting secondary database (`reporting.secondary`).

When the rebuild process has finished, the reporting secondary database replaces the primary reporting database (`reporting`).

Sitecore xDB does not allow rebuilding of the primary reporting database in-place. So the reporting secondary database is only required during the rebuild process.

The rebuild process is semi-automated but requires a system administrator to attach/replace databases in SQL Server and to make modifications to the xDB configuration.

2.5.3 Continuous Update of the Reporting Database

Continuous update of reporting database is the processing of interactions that have just ended. Unlike the *Rebuild of reporting database*, this is a continuous process that starts as soon as you launch Sitecore and which keeps the reporting database up to date with the most recent interactions, so long as the xDB is running.

How continuous update of the reporting database works:

1. The latest interactions are saved to the collection database.
2. Interaction data is added to the processing pool for aggregation.
3. An agent worker picks up the interaction from the processing pool and hands it on to an aggregator.
4. The aggregator pushes the interaction through the aggregation pipeline and converts the data into a form suitable for the reporting database.

The aggregation process converts data into a form that is easier to query and that is suitable for use with Sitecore reporting applications that use SQL Server.

Once the data is converted into the correct format, it is merged into existing reporting data that is stored in the reporting database, so the reporting database is updated (in sync) with the latest interactions on your website.

2.5.4 Rebuilding the Reporting Database

Rebuild of the reporting database is the re-processing of interactions that have already been aggregated into the reporting database for use by Sitecore reporting applications. To ensure that the latest changes to the collection database are reflected in the reporting database, you must rebuild the reporting database from time to time. When you rebuild the reporting database, its contents are overwritten.

Reasons for rebuilding the reporting database:

- After using the conversion tool to populate the reporting database (secondary) with analytics information from an earlier version of Sitecore.
- After having amended information in the collection database, in order to reflect the amendments in reports when looking at older data. Such amendment can be for example assigning channels to referring sites.
- If you have re-classified a search key word or traffic type, aggregated report data is not updated automatically.
- If the reporting database has been lost or is irrecoverably out of sync with the collection database, for example, due to a disaster or if the details of two contacts have been merged.
- In the *Executive Insight Dashboard* and other Sitecore reporting applications it is possible to reclassify data that has already been processed by the aggregation layer. This could cause the reporting database to become out of sync with the collection database.

Note
You only need to connect a second reporting database if you intend to perform *Rebuild of the reporting database*.

Note
All existing data contained in the reporting database is overwritten every time you perform a rebuild. However, for best results, we recommend you use a clean copy of the `Sitecore_Analytics` database every time you perform a rebuild.

2.6 Reporting

The Sitecore Experience Database provides a highly scalable and high performance solution for storing data and delivering reports. In earlier versions of Sitecore there was only a single analytics SQL Server database for all collection and reporting needs. For high traffic websites SQL Server is no longer scalable or high performance enough to perform all the data collection and reporting tasks now required.

To improve scalability and performance the xDB uses the following database systems:

- MongoDB (NoSQL) collection database - all website visitor data collected is written to the collection database.
- Microsoft SQL Server database – Sitecore reporting applications read data from a SQL Server reporting database.

Note
We recommend that you install the SQL Server reporting database on a separate server to achieve higher performance.

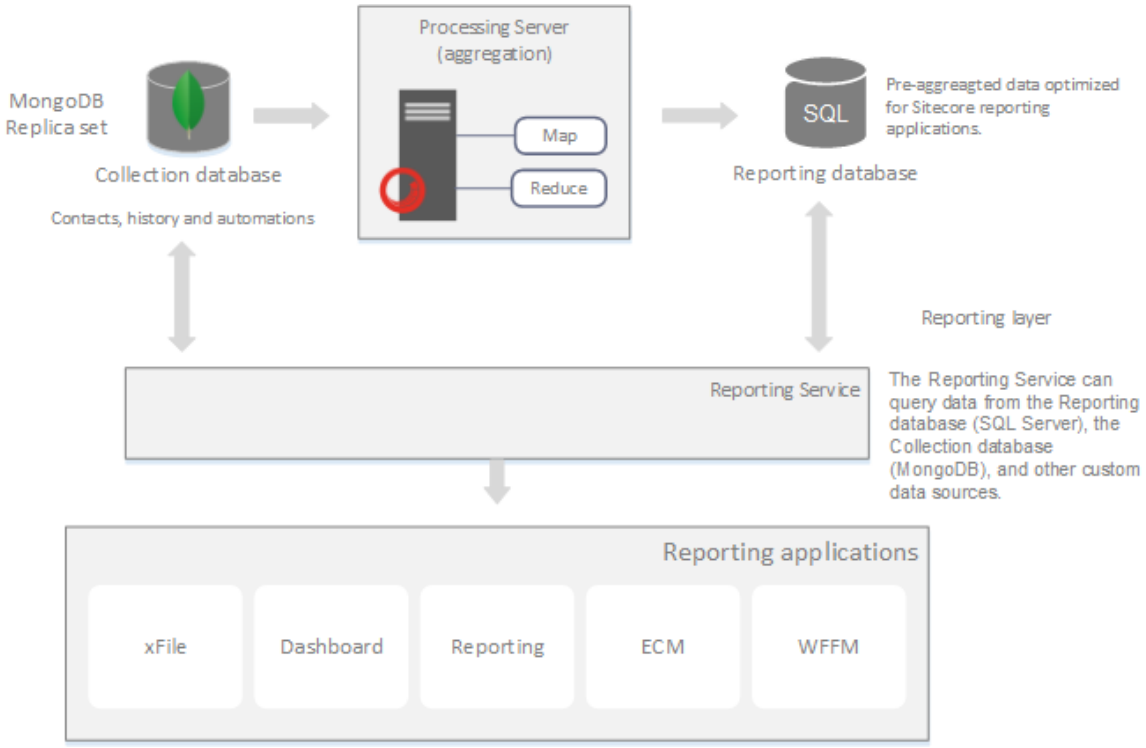
2.6.1 Reporting Architecture

The xDB collection database is a scalable, document-oriented database that can handle billions of interactions. It collects and stores all visitor data such as contacts, devices, locations, goals, engagement automation states and other details. By default it uses MongoDB to store all relevant information in documents which are roughly equivalent records in a relational database but quicker to query, so writing and reading is very fast. However, the data operations required for reporting are either not supported or only partially supported.

In the Sitecore Experience Database, the reporting architecture consists of:

- Reporting server – a Sitecore server that hosts the Reporting Service.
- Reporting Service – uses queries to fetch report data from various data sources such as MongoDB, SQL Server or other custom data sources such as a customer relationship management system (CRM).
- Reporting database – A Microsoft SQL Server database that contains *Fact* and *Dimension* tables of aggregated data from the collection database.
- Reporting applications – For example, the Sitecore® Experience Profile™ (xFile) and the Executive Insight Dashboard.

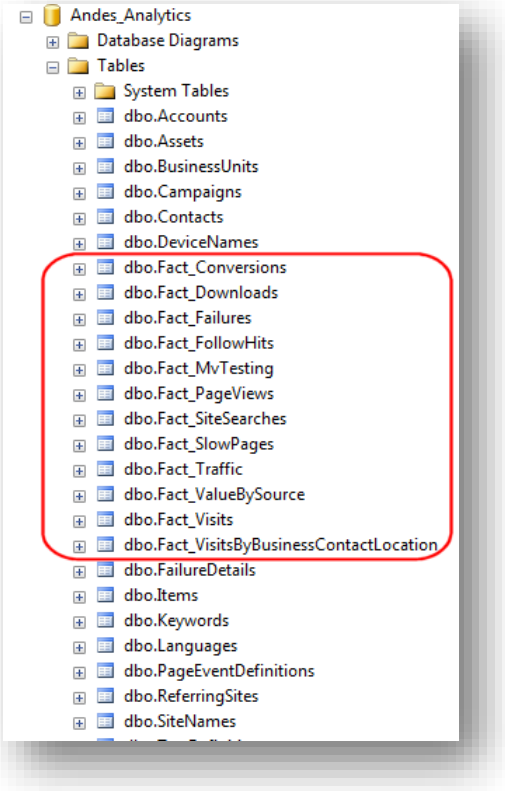
Sitecore xDB reporting architecture.



2.6.2 Report Queries

The reporting database consists of data specifically optimized for Sitecore reporting applications such as *Engagement Analytics* reports and the *Executive Insight Dashboard*. Pre-aggregated data is stored in SQL Server *Fact* and *Dimension* tables which help to improve performance by acting like a cache of the key collection data specific to reports.

SQL Server reporting database *Fact* tables.



However, despite the changes made to the reporting architecture, it is still possible to query both the SQL Server and MongoDB databases. This may be necessary as some visitor interaction data is only stored in the collection database and is not by default aggregated and passed on to the reporting layer to be used in the standard reporting applications. So it is possible to use the reporting database (SQL Server), the collection database (MongoDB), or both as the data source for report queries. You could even use custom data sources to retrieve information from other data systems such as a CRM.

To query the collection database (MongoDB) you need to use the reporting layer and the reporting API.

Sitecore reporting applications that use the reporting layer:

- Sitecore Experience Profile (xFile)
- Engagement Analytics reports (standard Sitecore reports)
- Executive Insight Dashboard (overview analytics data presented as charts and dashboards)
- Email Campaign Manager (ECM)
- Web Forms for Marketers (WFFM)

Chapter 3

Glossary of Terms

The aim of this chapter is to provide definitions for some of the key terms used in the Sitecore Experience Database (xDB).

This chapter contains the following sections:

- Glossary of Terms

3.1 Glossary of Terms

The term Sitecore Experience Platform (Sitecore XP) refers to the combination of Sitecore CMS, the Sitecore Experience Database (xDB), and Experience Marketing. The definitions and key terms previously used to describe the marketing platform remain largely unchanged but with the introduction of the Sitecore Experience Platform there are some new terms that we need to define.

3.1.1 Sitecore Experience Platform

The Sitecore Experience Platform is the combination of Sitecore CMS, the Sitecore Experience Database (xDB) and Experience Marketing applications.

3.1.2 Experience Database

The Sitecore Experience Database (xDB) is a set of content and digital marketing management features that you can use to interact and engage with contacts and customers. The xDB enables both customers and marketers to optimize the value of engagement.

The xDB collects all customer interactions from all channel sources in a central collection database, connects those interactions to create a comprehensive, unified view of the individual customer, and makes the data available to marketing in real-time for automated interactions across all channels.

3.1.3 Experience Profile

The Sitecore Experience Profile (xFile) is a Sitecore application that provides a single view of the customer in a visual interface for marketing.

The xFile enables you and your sales teams to monitor the key areas of customer experience and interaction, such as visits, campaigns, goals, profiles, automations, and keywords. For example, for each contact, you can see at a glance which events and goals they have triggered and how many engagement value points they have accumulated on your website.

3.1.4 Experience Marketing

Experience marketing is a term used to describe how you manage the different interactions with contacts and customers in an organization.

An interaction could be online, such as a visit to a website, a social media post, or opening an email campaign message.

The way in which you manage these interactions is very important. Effective experience marketing enables you to build valuable relationships between your contacts and customers, and your organization with the ultimate aim of creating lifelong customers.

3.1.5 Contact

A contact is a person that uses one or more devices to interact with your organization.

The contact is used as a container to store information about the behavior of your customers and potential customers from the devices they use, to all their online interactions, such as social network activity, email campaigns, and website visits. In fact, this system has the potential to record all interactions both online and offline. For example, it may be possible in the future to extend Sitecore to link to a CRM or ERP system containing all offline interactions.

In xDB, from starting out as an anonymous visitor to your website, contacts can be used to build up a picture of potential customers. By storing data from a wide variety of sources it is possible, over time, to build up a detailed picture of how a contact interacts with your organization.

In previous versions of Sitecore a contact was known as a visitor.

3.1.6 Device

A contact uses a device to interact with your organization. For example, a device could be a browser, smart phone, iPad, tablet, or something else. Details of the device are stored in the collection database in the device entity which is linked, along with the contact, to a specific Interaction.

One or more devices can be associated with a single contact and a single device could be associated with one or more contacts.

Note
Each time you access a website using a different browser this is classified as a new session and appears as a new device in the collection database.

3.1.7 Interaction

An interaction is the way in which a contact or customer communicates or engages with an organization. An interaction is usually a two-way exchange of communication and commitment that takes place via a specific media or channel. Two-way interactions usually require a higher level of commitment and trust than merely passively viewing a web page. They require a contact to provide something that identifies them, such as an email address, in return for something from your organization, such as access to a special offer or campaign.

Examples of online interactions:

- A visit to a campaign-landing page.
- A visit to your website that originated from a link on a social media site.
- Clicking *Like* on a social media campaign page.
- A visit generated from a link in an email campaign message.
- Opening an email that was sent from an email campaign.

All interactions from the first anonymous visit to customers showing higher levels of engagement and commitment are recorded in the collection database. This record of all online interactions is known as the *customer experience*.

In the xDB, currently the only type of interaction is an online visit but in the future this will be expanded to include offline interactions.

3.1.8 Online Visit

An online visit is a type of interaction in which a contact visits a website using a device.

3.1.9 Account

Account is a term used to organize contacts into groups usually based around a company or organization name. Account is identified by *Business Name* and *Country*.

Each Account has at least one Business Unit and each Business Unit is connected to an Account. Business Unit is identified by *Business Name*, *Country*, *Region,* or *City*.

3.1.10 Collection Database

The collection database (MongoDB) is a NoSQL database that stores contacts, history, and automations and is capable of storing vast amounts of customer experience data. The collection database contains information on contacts, devices and interactions, and is used as a repository for

the entire customer experience. This includes website visits, responses to email campaigns and social media interactions.

Data in the collection database can be categorized in the following ways:

- Customer experience data – for example, contact information and interactions.
- Marketing management data - How the marketer uses the xDB to manage the contact/customer experience – for example, it contains references to profiles, goals, campaigns, personalization, multivariate tests, and engagement plans.

Note
The collection database does not store the definition items for profiles, goals, campaigns and other Experience Marketing items that you have created, only references to them

By default the collection database is implemented as a MongoDB database.

3.1.11 Reporting Database

The reporting database contains aggregated data from the collection database which can be queried by the Reporting Service. The results of these queries are used by Sitecore reporting applications such as the *Executive Insight Dashboard* and *Engagement Analytics* reports.

By default the reporting database is implemented as a Microsoft SQL Server database.

3.1.12 Fact Tables

In the xDB, fact tables contain the data (the measurements and metrics) that you want to analyze.

Fact tables are similar but not the same as cache tables in the SQL reporting database that make xDB reporting data available to reporting applications such as the Executive Insight Dashboard, ECM and the iPad Dashboard.

3.1.13 Dimension Tables

Dimension tables are companion tables to a fact tables. They contain descriptive attributes or textual fields that help you understand and analyze the data in fact tables.

Dimension tables are similar to lookup tables in the SQL Server reporting database. SQL Server uses a script to update the dimension table with data from fact tables.

3.1.14 Aggregation

The aggregation layer groups and reduces data from the collection database and then stores it in the reporting SQL Server database. Specifically, this means that data is processed into a form that can be easily queried by the Reporting Service for applications, such as the *Executive Insight Dashboard and Engagement Analytics* reports.

By default, the aggregation server is a Sitecore CMS server with the client removed.

3.1.15 Rebuild of the Reporting Database

The rebuilding of the reporting database is a process that is performed after you have used the conversion tool or after you have re-classified a search key word or traffic type.

Note
All existing data contained in the reporting database is overwritten every time you perform a rebuild.

3.1.16 Reporting Service

The Reporting Service queries your databases and sends the results back to Sitecore reporting applications.

Queries can retrieve data from both SQL Server and NoSQL databases. The Reporting Service queries SQL Server for grouped data, aggregated data and trends. It queries the collection database for individual contact or interaction data. The type of query you create, depends on the type of data that you need for the reporting applications you are using.

3.1.17 Session State Server

The session state server is a database that contains all the information required to serve a session, for example, identifying the contact, previous behavior of the contact, characteristics of the contact and so on.

By default, all session state information is stored in the collection database (MongoDB) running on the content delivery cluster.

3.1.18 Private Session State

Private session state holds, for example, the visit information with pages visited and page events triggered. It holds all the information relevant to the individual visit/interaction.

Private session state includes:

- Visit details
- Pages
- Page events

Note
Private session state is the essentially same ASP.NET session state.

3.1.19 Shared Session State

Shared session state is contact and device information (and associated information) and can be shared among multiple sessions on the same server or cluster.

Shared session state includes:

- Contact
- Device
- Engagement states