CX Company

DigitalCX Security Document

Policy Documentation

Created on	30th June 2015
Author	Jeroen Brouwers
Version	3.2
Label	Public Restricted

CONTENTS

Revisions

Date	Version	Author	Description
01-03-2015	0.5	J.Brouwers	Initial Version
01-06-2015	0.6	G. Willems	Reviewed Version
30-06-2015	1.0	J. Brouwers	Final Version
01-07-2016	1.1	P.Pelzer	Yearly update; full review; addition of PEN tests and EU data
01-09-2016	1.2	J.Brouwers	Yearly update
01-10-2017	1.3	J.Brouwers	Yearly update, small changes
01-01-2018	1.4	J.Brouwers	Admin Portal updates
03-07-2018	2.6	B.Dirks, D.d'Hauwer, F.Orlandini	Yearly update; GDPR; ISO; Platform Updates
13-05-2019	2.7	T.Bullock	Update ISO information
04-10-2019	2.8	T.Boormans	Yearly update; full documentation review
10-10-2019	3.0	T.Boormans	Extending documentation and updating structure of document
07-11-2019	3.1	T.Bullock	Converting to house-style
26-05-2020	3.2	D. Beckers	Updated engine archt. diagrams

Intended Audience

This DigitalCX Security document is intended for security professionals, in order to gain further insights into how DigitalCX is set up, and secured.

Introduction

DigitalCX is CX Company's automated customer engagement platform. We develop in-house and fully cloud. DigitalCX allows content editors and developers to work together to

create use cases ranging from FAQ management to a fully conversational process handling chatbot.

This document provides an overview of our hosting environment, the setup of DigitalCX, and the security measures we take to keep our client's data safe.

Hosting Platform Security

Hosting Platform: Azure

DigitalCX is hosted on Microsoft Azure. Azure offers a large range of services of which only a few are used. This will be discussed in the following section. Microsoft Cloud complies with a range of standards.

Microsoft offers different regions and these regions comply with local regulations.

GDPR

GDPR states that managing personal data is bound to the EU and should not be exported outside of the EU. More generic information regarding GDPR can be found online via the European Commission website.

This is the official statement from Microsoft Azure : "The GDPR imposes new rules on companies, government agencies, nonprofits, and other organizations that offer goods and services to people in the European Union (EU), or that collect and analyze data tied to EU residents. The GDPR applies no matter where you are located.". The official CX Company

statement states that the company complies with GDPR and how personal data is processed.

ISO 27001

ISO 27001 is an information security standard that is intended to bring information security under management control. CX Company has been ISO 27001 certified since 2018.

Azure Services

DigitalCX uses the following services from Azure:

Azure service	Description
App Services	The majority of elements are deployed as web apps
Virtual machines	The majority of databases and reporting portal are deployed on dedicated virtual machines (VM)
Virtual network	Access to the management portal and direct access to the VM is restricted via VPN access from our DevOps location
SQL database	Azure SQL database is used for the master database
Redis Cache	Redis is used by the DigitalCX engine for session storage
Queue	Azure queues are used for temporary async logging
Scheduler	Azure Scheduler is used for data publishing jobs
Service Bus	Service bus is used as a logging queue
Storage accounts	Azure disk storage with extended capabilities for access and queued processing
Load Balancers	Azure appliances for realizing high-availability environments inside the cloud
Recovery Services vaults	The Azure recovery vault is used for backing up the VM
Traffic Manager	General service for routing traffic
Azure B2C/AD	Azure Active Directory is used for identifying and access management

The main security setup is detailed below:

Azure Security Measures

Azure itself has a long list of security measures that apply for all customers, including IDS and DDOS. DigitalCX benefits from that. On all virtual machines of our platform, TrendMicro's Deep Security is installed **as malware and IDS protection**. Microsoft provides extensive documentation on how to set-up proper security as well as a Best Practice Analyzer to detect common security problems from settings. Azure provides a personalized cloud consultant system to help following the best practices which is called Azure advisory. Our DevOps team follows these best practices in order to keep the DigitalCX infrastructure safe.

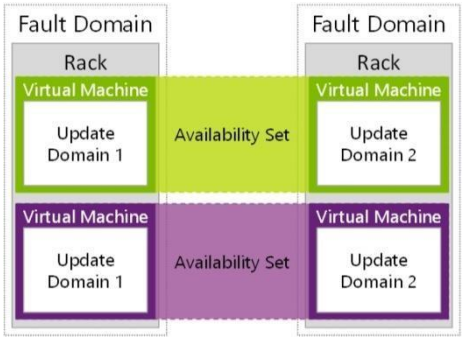
Azure AD B2C – authentication

DigitalCX uses Azure B2C for user identity and access management. It is an extension of Microsoft Azure Active Directory. Azure B2C is also used for service-to-service identity internally within the platform. **Oauth2** is used throughout for all authentication, ensuring claim based security on each internal and external endpoint of our platform that has the ability to write.

User handling itself is described in a later section.

Azure Uptime and Recovery

Azure provides a public SLA stating that all their services have at least 99.9% availability, their current uptime is publicly available. DigitalCX's SLA states the same availability for all its production services.



Using a high availability setup and geo failover increases the availability of the system. Azure Virtual machines have the option to be set in an availability set to reduce downtime and increase the availability

during Azure updates. Automatic daily backups ensure limited loss of data in case of catastrophic environment failure as well as easy restore.

The mentioned setup is used to maximize uptime and minimize downtime in case of disaster this is discussed in the data backup and restore section.

Virtual Network

Our platform is exposed via REST APIs. All services of the DigitalCX platform have APIs but only a limited amount are exposed. The rest is safeguarded with no attack service via a virtual network. Our DevOps team has access via a VPN from our Maastricht Office.

Regions

Azure provides regions of deployment around the world. Each region has multiple data centers that are in constant sync and failover to minimize downtime.

DigitalCX is currently deployed only within **European datacenters only**. DigitalCX uses the Azure regions North and West Europe. Data does not leave the EU (see also GDPR). Both data centers hold the same setup for the engine and are in active failover. Through elastic scaling, each region can take over all traffic from the other.

We can scale out to other regions – on demand – if required.

DNS

DigitalCX makes use of the protected, scalable DNS infrastructure of Cloudflare. Using Cloudflare's Anycast platform we have a world-wide local DNS presence which makes us less vulnerable to attacks and faster loading times for our web applications.

On at domain level we strive to enable DNSSEC protection to solve the risk of man-in-the-middle attacks. This layer on top of DNS assures us that our customers only receive data from us, and send only data to us, even if there would be an attacker in the middle of our connection (such as can be an access provider or a governmental organization).

Traffic Management

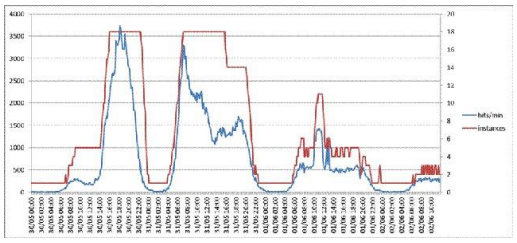
Azure Traffic Manager enables us to control the distribution of traffic across DigitalCX endpoints.

Traffic Manager provides some key benefits:

- Distribution of region based traffic
- Continuous monitoring of endpoint health and automatic failover when endpoints fail

The most important point to understand is that Traffic Manager works at DNS level. Traffic Manager uses DNS to direct clients to specific service endpoints based on the rules of the traffic-routing method.
Clients connect to the selected endpoint directly. Traffic Manager is not a proxy or a gateway. Traffic Manager does not see the traffic passing between the client and the service.

Scaling



DigitalCX uses has the Azure functionality to auto-scale App Services. This effectively means that when more traffic comes into the DigitalCX platform, the hardware automatically scales up to handle the load. This helps prevent any downtime or problems when traffic peaks suddenly. DigitalCX uses auto-scaling for the App Services in order to maximize availability for all components of the platform.

VMs typically don't need to scale with extra traffic, but they can automatically scale to a certain degree based on metrics like CPU and IOPS. If more resources are needed structurally, this is handled by DevOps based on the monitoring data.

Data Backup and Restore

All virtual machines are backed-up daily. The majority of virtual machines contain the databases and the reporting cubes/dashboards. If a backup fails DevOps will be notified and the problem will be resolved within 24 hours. The backups are created with Azure tooling and are stored in an encrypted storage account. No offside transport is done of any kind. On top of that for all client databases the latest 10 databases are stored, older databases will be removed automatically.

Business Continuity and Disaster Recovery

DigitalCX has an active-active setup for its engines in two geographically separated regions. Each region can scale elastically to handle all the traffic needed. This way, as well as the measures mentioned above, the chance of a full outage is limited.

To keep our systems online whenever during a disaster, CX Company stores playbooks of the datacenter and server setup. These playbooks can be executed on an environment in another separate datacenter region in order to get systems back online with minimal effort.

In cases where data recovery is needed, our daily backup can be restored quickly with an expected restore time of 4 hours. Ensuring full data synchronization of older data can take up to 72 hours, yet in the meantime all components are already working.

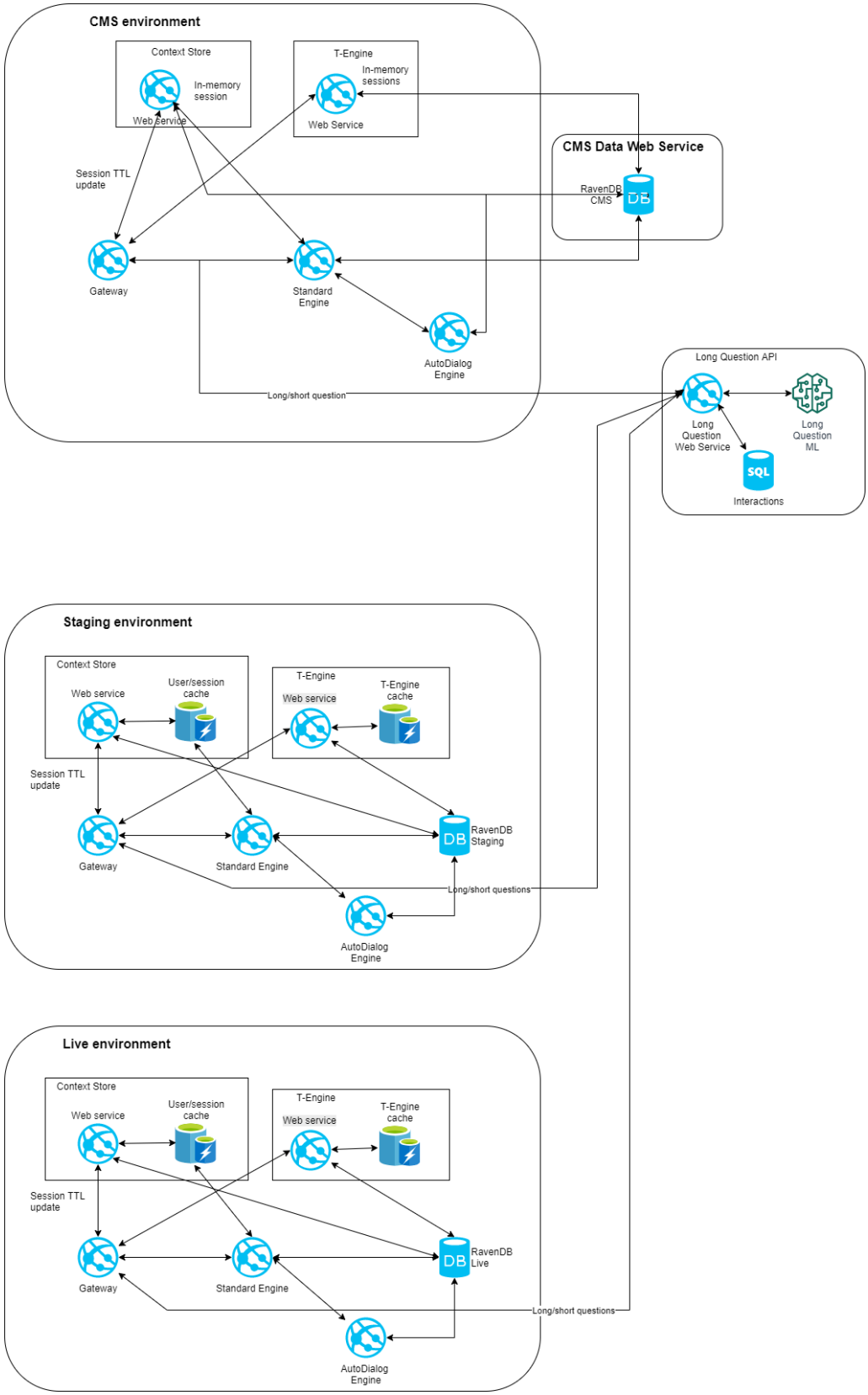
DigitalCX Architecture

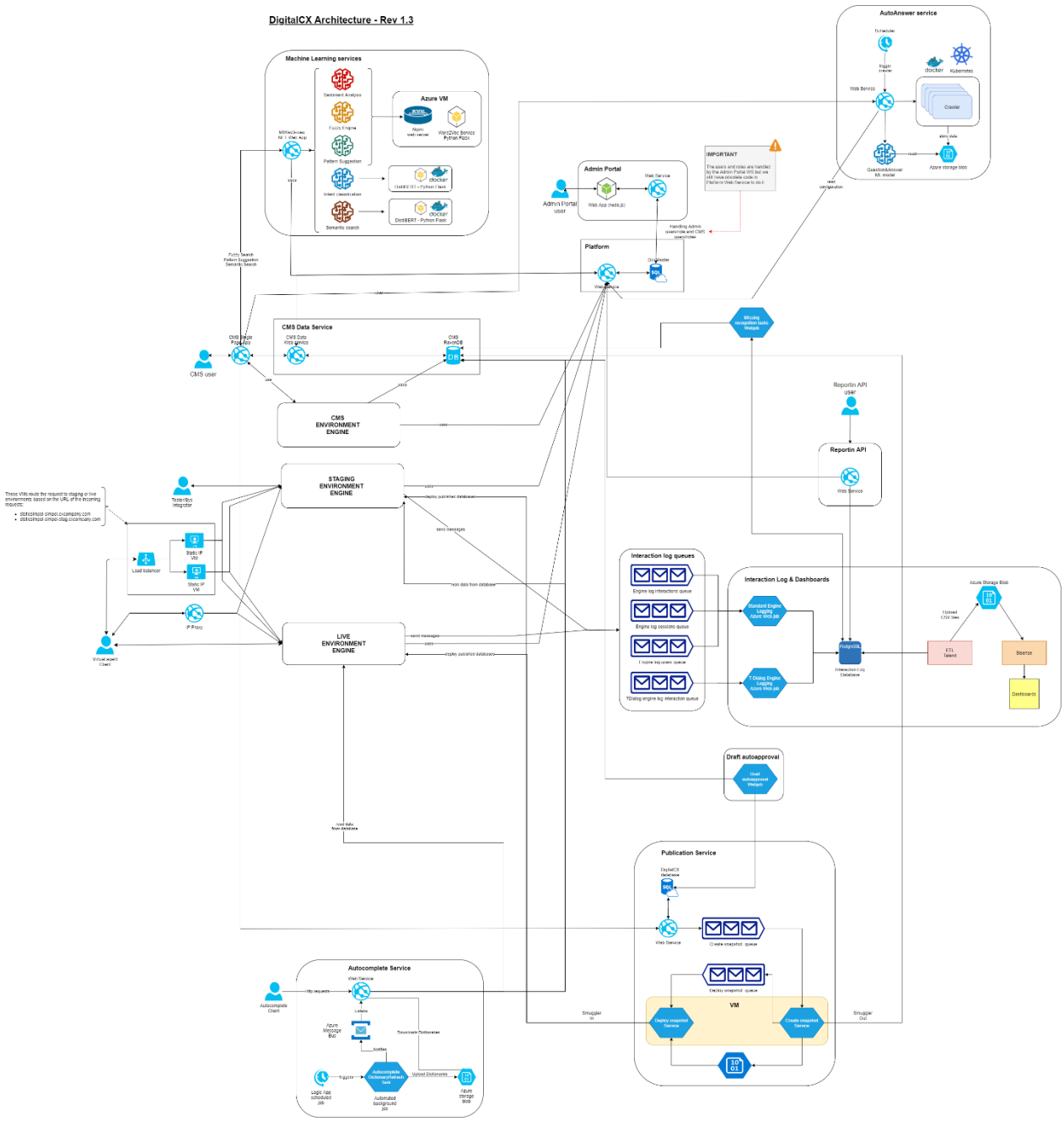
DigitalCX is CX Company's platform for automated customer engagement. It is fully built for and deployed on Azure, with all the security measures mentioned in the previous chapter.

Introduction

DigitalCX is a multi-tenant application with client data isolation, e.g. each client has its own data repository. The following diagram shows the different parts of the platform.

DigitalCX Engine Architecture - Rev 1.3





DigitalCX consist of the following parts:

- **Development** is about setting up the project's knowledge bases, and Quality Assurance. It's where the ***CMS*** is deployed as well as the Engine for testing. Additionally, this is where the automated content QA jobs are running and the

Package Manager that controls the data flows from dev to staging and production, as well as the backups.

- **Staging** is solely for (acceptance) testing and can be used by a separate testing platform, a content testing environment, or whatever is needed for approval before deploying it for production. Only the engine is deployed here.
- **Production** is where DigitalCX is deployed, mostly integrated in a website or app, for use by the end-users. Only the engine is deployed here.

Additionally, there is the **Reporting** part that is integrated into the CMS. There is a separate **Admin Portal** for each client.

DigitalCX Engine

The DigitalCX Engine is the main software process that handles the user input from end-users, provides the integrated natural language processing engine, as well as the search algorithm to find the best possible output to that user input. Basically, it's the engine that answers the open questions, handles the flow of a dialog, and provides the automated responses to user actions that together form the conversation.

The engine is a multi-tenant application that connects to a client specific databases. That database stores the outputs that are used for the conversations. The database is administered by the Content Management System (CMS).

Every customer can have multiple projects. Each customer and each project will be stored in a separate database with a unique randomly generated API key to restrict and separate the access from the Data Access Layer of the CMS.

DigitalCX Engine Naming Conventions

DigitalCX uses the following URL convention:

Dev: *https://dev.digitalcx.com/[CustomerKey]/projects/[ProjectKey]*

Staging: https://[CustomerKey]-[ProjectKey].staging.digitalcx.com
Production: https://[CustomerKey]-[ProjectKey].digitalcx.com

So, when Acme company creates a project called 'TestProject', the following CNAME's are mapped:

https://acme-testproject.dev.digitalcx.com/ https://acme-testproject.staging.digitalcx.com/ https://acme-testproject.digitalcx.com/

Each will have their own API key with settings to control access and throttling. Please refer to the DigitalCX Engine API Documentation for full specification on URLs, API keys and (required) query parameters

DigitalCX CMS

DigitalCX comes with an elaborate Content Management System (CMS) to allow content editors (e.g. users of the CMS) to create, change or delete parts of the automated conversation. Content editors can have different roles that are managed through the Admin Portal, and enforce access to a client database (always within the scope of a customer), as well as the access right within the CMS (e.g. different functions, e.g. RBAC).

The data in the client database can only be manipulated via the CMS portal. All access attempts from users to the CMS portal are audit logged, successful or failure. All changes to the content within a client database are stored as named revisions. Changes to the content can only be made to the CMS system and then published to the staging or staging and production environment as discussed in the flow of changes.

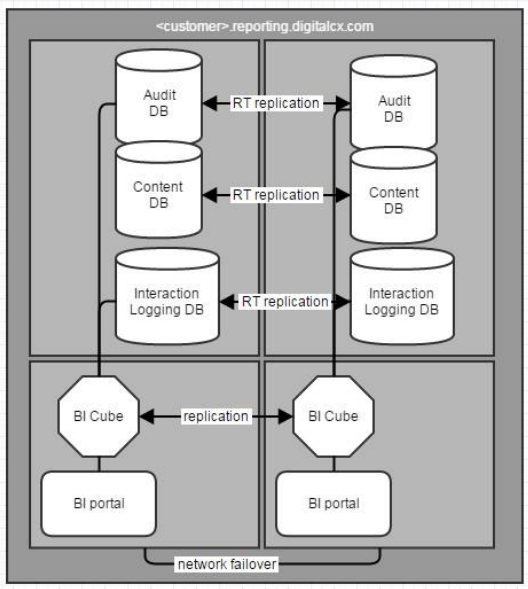
DigitalCX Reporting

The reporting segment of the platform provides the aggregated statistics on the end-user's interactions with the Engine. The access rights of the content editor in the CMS are copied over to the reporting segment to ensure strict data access, also on the Application Logs.

The reporting segment provides Dashboards to content editors to learn about the end-user interactions as well as reports from the learning algorithms which provide suggestions for improvement to the client database, based on those end-user interactions.

Technically, reporting is separated from the regular environments and located on VMs in the Western Europe region only. DigitalCX uses PostgreSQL as its logging database platform, for all services, for all regions, for all logging types except debug and tracing where we use Stackify APM.

In PostgreSQL, there are separate databases for logging Interactions, Errors, and Audit information. Logging



to PostgreSQL is only possible from a pre-set list of apps (e.g. only our own services across the world are whitelisted) using the persistent Azure Service Bus as a queue, and an Azure Worker Role to gracefully insert the logs.

We've chosen SiSense as a reporting environment, to efficiently handle the data in cubes and dashboards. The reporting environment is part of the CMS and cannot be accessed separately. The cube updates frequently to represent the latest data.

DigitalCX Admin Portal

The DigitalCX Admin Portal provides customer administrators with a separate interface (e.g. not part of the CMS) to manage the projects within the scope of that customer. Practically, the Admin Portal provides Administrators with the options to create and delete projects, create and delete users, set change or delete the user access rights to one of the projects, and set generic project settings on publishing and workflow.

DigitalCX works with a Master-Client approach where the Master database holds the references and settings for the client database. The Master database is completely separated (different virtual machine) from the Client. All services call the Master database to retrieve the project and user related settings and cache these settings for 15 minutes. This results in a small delay for the distribution of changes.

The user interfaces are built as an SPA with a NodeJS back-end to implement the additional security layer to access the data. Connections are SSL/TLS only. Read access from any service that is part of the DigitalCX system is routed via s separate endpoint for which every call is audit logged and access is whitelisted.

Users for the database are as defined in the chapter User Handling. Depending on the level of access various settings are available to control the environment on each level, general settings for entities one level down, and management of users one level down.

Administrators are specific to one customer entity. Administrators cannot have access to the client projects (via the CMS). Administrators have to login with 2FA. As throughout the platform, login attempts are logged as well as any revisions to the data within these data containers.

Digital CX Operations

DigitalCX is managed by CX Company with a dedicated in-house team. The team consists of the Information Security Officer, the CTO, a Development team, a DevOps team, and a Support team. Given the nature of their functions and close workings with the systems that handle our customers' data, they have (criminal) background checks, proper training and evaluation.

This section details each team with respect to the security of the DigitalCX platform.

Information Security Officer

Within CX Company, our Information Security Office (I.S.O) is also DPA, and involved in the platform management to support the security-by-design of the platform, and handle incidents.

All Incidents, both operational or security related, are logged according to processes as defined within ISO 27001. The I.S.O is kept up to date on all security problems as well as any CIA problems with the platform. Weekly meetings and monthly reviews ensure a constant focus on stability and safety of the platform.

In this way the Support, DevOps and Development teams have extra insight into what happens on the DigitalCX platform, allowing them to see where improvements are needed and what the customers experience with the platform.

DigitalCX Development

DigitalCX has its own dedicated development team, fully in-house at CX Company's Maastricht location.

DigitalCX code

The entire DigitalCX back end system is built on Microsoft .NET technology. DigitalCX runs entirely in the Microsoft Azure Cloud. The coding process follows the best practice enforced through Visual Studio integrated coding standard as well as the SonarHQ.

The DigitalCX CMS is built on top of the Durandal JavaScript framework. It uses Gulp as a build tool along with Mocha, Karma, Chai and Sinon for JavaScript testing. The DigitalCX Admin Portal is built on top of the VueJs framework and served through a Node.js based server application that exposes a GraphQl API alongside a RESTful API. Its front end is built through the use of Webpack 4 and tested with Jest. The Admin Portal is hosted in a Node.js server. The Node.js server code is also tested using Jest.

The entire codebase, for both front end and back end, is in Azure, stored in Git in Visual Studio Team Service (VSTS).

Coding process

The code is covered by unit tests, based on NUnit framework and Moq as mocking library, complemented with integration tests, built using a SpecFlow framework. The continuous build server, on each commit of code, automatically build the code, checks the unit tests and reports to the entire development team if the build fails or if any unit test does not pass.

The Development Team follows the agile philosophy, using a Scrum methodology. The development is broken down into iterations (Sprints) of 3 weeks each. Every sprint is planned upfront, defining the features that the team is going to release. The features are broken down into smaller tasks. The tasks are stored in Jira. The quality of the code is enforced in the process using pair programming and code reviews, using pull requests.

When storing new code to our repository the development team works in a way which makes code fully trackable, thus allowing us to be able to identify the responsible code and individuals for any root cause analysis.

Development uses a separate, fully synced copy of the Production environment which our customers use, for testing before releases.

Software release

After the sprint ends, the team starts the deployment phase to move the new feature into Production, in collaboration with the Support and DevOps team.

The Support team prepares a test plan and starts testing the new features and changes. Customers are informed about the release and possible changes of the scope or date as decided during the testing phase.

The elected release manager works with DevOps to create the Release Checklist. When Support signs off on the release, the Devops team prepares the deployment of the release and deploy the code in production, executing data migration or any other steps needed.

When the deployment completes, the Support team checks the new features and changes in Production. If everything works, they sign off the release and they inform the customer about the finalization of the release. If not everything works as expected then they inform the release team. If there is a critical error inside our production environment then a rollback of the release is being executed, and a hotfix is being created at a later moment to be able to do a successful release in a later maintenance window.

Automated Deployments

DigitalCX is deployed using Visual Studio Team Services. VSTS not only automates the build, testing and deployment of DigitalCX, it gives us complete traceability to see everything in the build including changes to code, reviews, test results, and code quality analysis reports.

Release pipelines allow us to automate deployments across multiple environments and always be in control of promoting and deploying releases by using approval workflows. All configurations are handled as code and are documented and saved under Source Control.

Backward compatibility

There are two different type of release:

- Minor releases
- Major releases

For minor releases, the backward compatibility is always enforced. In case of any change in the data structure, the release will include a data migration process to reshape the data structure.

For major releases, full backward compatibility might not be supported for all the DigitalCX components and interfaces. In this case, the Support team includes the breaking changes in the communications sent to the customers at least 2 months up front. In any case, the previous version of DigitalCX continues to be supported for at least another 3 months to accommodate the changes for all customers involved.

Software dependencies

DigitalCX uses a list of dependencies as part of its software. These dependencies are incorporated in the security audits. For each dependency, the DevOps team and the Security department keeps track of any known or newly reported security problems to be able to investigate the issues in our implementation and where needed change the code to mitigate the risk. Appendix II holds a complete list of dependencies.

DigitalCX DevOps

Monitoring

All components of DigitalCX are monitored continuously. The monitoring system consists of several monitoring features:

- App Performance Monitoring
- Code level performance profiling
- Usage and performance of all application dependencies
- Error Tracking

All monitoring systems, partially from Azure, partially from Stackify (.NET apps) and KeyMetrics (NodeJs apps), provide automated alerts to DevOps. Access is linked to the AD. Incidents are handled according to incident processes and thus reviewed, handled, escalated where needed, and reviewed periodically.

ITIL

DevOps works in line with ITIL best practices. This includes maintaining an up to date inventory of all assets, keeping track of service levels, capacity planning, etc. More information is available online.

Patch management

Operating System patches are automatically installed on the Development and Staging environments. When patches are installed a standard set of tests is performed after each patch update. Given no significant changes, patches are automatically deployed on all VMs the day after. DevOps can intervene to prevent these patches from updating the Production environments would any issues arise from the tests.

Software patches of dependencies (like RavenDB, SiSense or PostgreSQL) have a different regime with full unit-testing of all the related components. The Support team tests all functionality affected by updates on the Staging Environment before a Software patch is approved for deployment to production.

Yearly audit

DigitalCX is audited annually on all of our network equipment and software components during an extensive OWASP security check (pentest). During this check-up a team of security engineers audits our documentation and all our application interfaces to make sure that unwanted access and privilege escalations are impossible, that there is adequate auditable logging and that procedures are being followed. OWASP works great on top of our ISO 270001 certification in which we opted in for only granting access using the “least access possible” principle.

DigitalCX Support

DigitalCX Support handles user communication, incoming tickets, release testing, and the DigitalCX Support Portal. The team has a hotline for critical issues of the platform outside our business hours to ensure 24/7 support.

Data Operations

Handling Personal Identity Information (PII)

DigitalCX takes an explicit approach to any data entering the system because PII definitions vary across companies, verticals and legislation of different countries.

For each project a whitelist is defined for accepted user properties. These user properties are provided by the calling request, query connection string, the header, or an external source (part of the engine setup). Any input values outside of this whitelist are ignored in the interface and are not logged anywhere.

The allowed input can be set to be used for one or more derived values. This is done by using JavaScript functions that can be a part of the project. The allowed input can be set to be used for internally search through DigitalCX's knowledge base, and/or also as an input for answers, and/or not for logging.

DigitalCX provides standard UI masking rules to prevent personal data from end-users entered in questions or feedback to be logged in the application logging. Default UI masking rules apply to all projects will remove email addresses, IBAN, passport numbers and credit card numbers. Also, numbers with 4 digits or more than 5 digits are removed. Tailor fit masking rules can be applied when required.

Access to the logs is restricted and audited.

Flow of Data

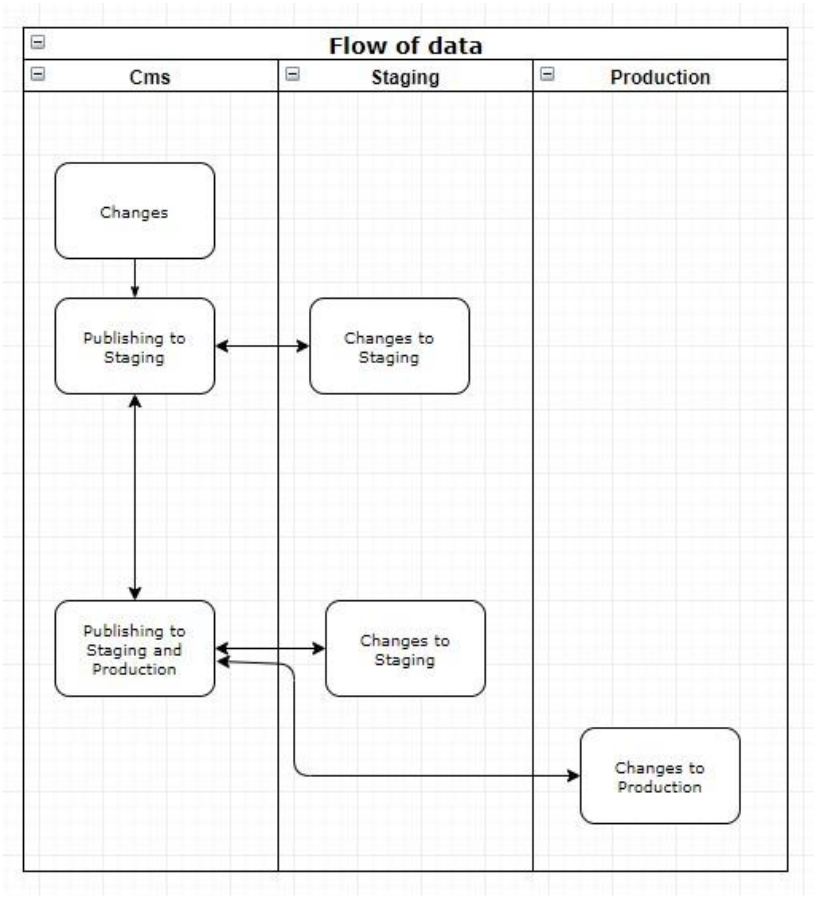
There are two major flows of data:

1. The flow of changes (publishing)
2. The flow of process data (reporting)

The flow of changes (publishing)

This data entails settings for the working of the system. The application allows to the customer to test changes in a Dev/CMS function before applying the changes to production. The changes are made in the CMS environment first. The changes can be published to

Staging. This is a similar environment as Production were the changes can be validated. After that the changes can be pushed to Staging and Production.

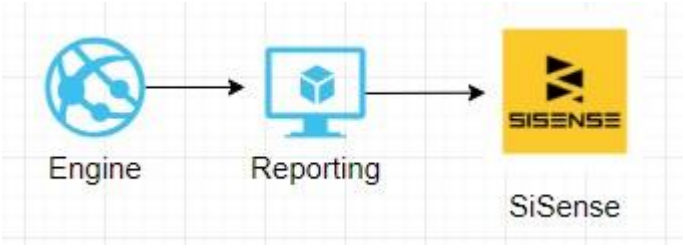


The data is always published to staging. By default once a day all updates from dev/CMS are pushed to staging and production. That setting is controlled in the Admin Portal. For each customer and project, this setup can change. Additionally, content editors with the correct permissions are able to publish on demand.

When publishing, the data will be validated. If faulty the previous database is restored automatically. Notifications will be sent out and the source of the error will be determined. Databases in staging and production are read-only.

The flow of process data (reporting)

This data entails the masked interaction data from the system. Customers require data to analyze and improve the working. Masked reporting data is collected and logged in a Postgresql database system for all services, regions and logging types except for debug and tracing. For debug and tracing we use the Stackify APM.



The reporting data is being processed and aggregated by a separate VM (the reporting server) and is prepared to be sent to the SiSense cube. The aggregation is done by project and by hour.

The reporting data will be used in SiSense to present the data in cubes and dashboards. The reporting environment is part of the CMS and cannot be accessed separately. The cube updates multiple times per day to give the customer near real-time data.

User Handling

User Security Model

DigitalCX applies the Least Access Possible policy inside the organization to prevent employees from making mistakes in areas other than that in which their profession lies. Combined with the audit logging makes this a very trustworthy model for managing user roles and rights.

DigitalCX's security model is build up in several layers where each layer is isolated from the next. For Project and Customer administration, an admin at any level can only add or change attributes from that level and create users one level down. DigitalCX DevOps has the Sys Admin role thus can never access a project's data.

Table 1 - User security model

	Sys admin	Partner admin	Customer admin	Security officer	Project Admin	Project User
Sys admin	X	X				
Partner admin			X			
Customer admin				X	X	X
Security officer						

Project admin	X

User Roles

Within a project, there are several types of users. The following table shows the DigitalCX user roles:

Table 2 - DigitalCX user roles

Project user role	Description
Content team lead	In charge of content editors in a leading and controlling role
Content editor	Makes changes to the content of the knowledge base
Business intelligence designer	Creates/modifies project specific Dashboards
Business intelligence user	Dashboards user
Quality assurance analyst	Quality Assurance on content and user interactions
JS Developer	UI designer and web hooks developer for DigitalCX

A Customer Admin can create new users; a Project Admin can assign them to a project. This way, a customer can have multiple projects, each operated by roughly the same team of people, where the Project Admin defines the finer grained access these users have.

You can find the default security settings of each role in **Appendix I: User Roles and Security settings**.

User Creation Flow

In DigitalCX, the flow to create a CMS user is the same as the flow to create an Admin Portal user. In both cases, an Admin Portal user has to create a user, inserting username and an email address, and assign the relevant user roles. DigitalCX automatically sends an invitation via email to the new user. The user is now in a pending state and cannot enter any part of the system. In the email, the invited user can find an explanation on the steps to follow to finalize their account creation. The user has to:

- Click on a confirmation link, to confirm the account
- Choose a password that has to match the set password policy rules (strong typed password)

It is important to notice that the confirmation link specified in the email has an expiration period, after a specific amount of time the confirmation link is no more valid. If the confirmation link expires then the pending user, it is automatic discarded and the Admin user is notified.

For the project users that finished this procedure, the Project Admin is notified so that they can assign access to one or more projects within the scope of their permissions , after which the user account and the access rights are set thus allowing a user to log into the CMS.

It is important to remember that an Admin user can only create new users on the level they are on or one level down (with fewer privileges).

Disabling Users

An Admin can completely disable users one level down, and choose whether these users are notified of this (or not). An Admin can also mark users as compromised thus forcing a repeat procedure of the user creation.

Compromised users are currently only marked manually based on audit logs that an Admin can access for their scope. Typically, if the Master database or part of it, is compromised then the DigitalCX DevOps team will notify our customers at first notice to help them address the potential damage. Part of this procedure might be to lock down the Master Platform for some time to isolate the attack and its impact, to make sure no further damage can be done.

Appendix I: User Roles and Security Settings

	Sys admin	Partner admin	Customer admin	Security officer	Project Admin	Project User
Sys admin	X	X				
Partner admin			X			
Customer admin				X	X	X
Security officer						
Project admin						X
Content team lead						
Content editor						
BI designer						
BI user						
QA analyst						
JS Developer						
	Sys admin	Partner admin	Customer admin	Security officer	Project Admin	Project User
Set Language allowed			X			
Set NOC info			X			
Set security info			X			
	Sys admin	Partner admin	Customer admin	Security officer	Project Admin	Project User
Normalization overrides			X			
User password policy				X		
Module enable/disable			X		X	
Customer IP-range				X		
Allowed region usage				X		

Backup data encryption				X		
Allow cookies on Engine				X		
Allow cookies on CMS				X		
2-factor authentication rules				X		
Privacy data handling				X		
User types CRUD operations		X	X			
IP range restriction for the Engine				X	X	
Project user entity CRUD			X			

Invite user to project					X	
Set min/max numb. **Backups**				X		
CRUD backup schemes					X	
R current backups			X		X	
Restore backup					X	
Region usage					X	
Data publish schemes					X	
User tasks	X	X	X		X	X
CRUD Q&A						X
CRUD Event for Answer						X
Create Events					X	
CRUD Patterns						X

CRUD Classifications						X
Master DB backup restore	X					
Master DB backup schemes	X					
Use test console						X
Use dialogs						X
Use dashboards						X
R system errors	X					
R interaction errors			X	X	X	
R audit logs				X		
R interaction logs			X	X		X
Set log masking rules				X		
Set log encryption rules				X		
Set Redis cache encrypt key	X			X?		
Set logging queue whitelist				X		
Set log encryption key				X		
Create new project			X			
Set allowed languages			X		X	
R data versions in regions			X	X	X	

Appendix II: Software Dependencies

Any dependencies on external software packages or libraries are regularly reviewed for updates. This to, at least, ensure stability and security of the services involved.

DigitalCX's architecture relies on the following software from Azure

- Azure Storage account
- Azure Web App (should we include these? They are not the maximum about security)
- Azure Logic Apps
- Azure Redis cache - any user data are encrypted.
- Azure Service Bus - messages over the bus do not hold any sensitive data
- Azure SQL Server
- Azure Stackify extension - application error tracing validated to not hold sensitive data
- Virtual Machine – with encrypted disks

In addition, DigitalCX's Virtual Machines use:

- TrendMicro Deep Security for Intrusion Detection
- Antivirus from BitDefender.

DigitalCX's user interface uses the following libraries:

- Durandal JavaScript framework
- Gulp
- Mocha
- Karma
- Sinon-Chai
- VueJs framework
- Node.js
- GraphQL
- Webpack 4
- Jest
- Stackify client

DigitalCX's CMS and engine uses the following dependencies:

- Microsoft .NET 4.7.1 framework
- Microsoft .NET core 1.1 and 1.2
- Microsoft.IdentityClient - client to communicate with Azure B2C
- Autofac - dependency injection component.
- AutoMapper - converts an object of any type to another object.
- NLog - logging handler.Security Documentation DigitalCX Platform
- Raven Client - client lib to connect to the RavenDB databases
- PostgreSQL NpgSql - client lib to connect to the PostgreSQL databases
- NUnit
- Polly to handle the error/retry in code
- SpecFlow for integration testing
- Python
- Docker

DigitalCX's database server uses the following dependencies:

- RavenDB
- Azure SQL Server

DigitalCX's reporting uses the following dependencies:

- SiSense
- PostgreSQL