



# Viz Social Deployment Guide

Version 1.0



# Viz Social

Powered by [Never.no](http://Never.no)



**Copyright © 2021 Vizrt.** All rights reserved.

No part of this software, documentation or publication may be reproduced, transcribed, stored in a retrieval system, translated into any language, computer language, or transmitted in any form or by any means, electronically, mechanically, magnetically, optically, chemically, photocopied, manually, or otherwise, without prior written permission from Vizrt. Vizrt specifically retains title to all Vizrt software. This software is supplied under a license agreement and may only be installed, used or copied in accordance to that agreement.

### **Disclaimer**

Vizrt provides this publication “as is” without warranty of any kind, either expressed or implied. This publication may contain technical inaccuracies or typographical errors. While every precaution has been taken in the preparation of this document to ensure that it contains accurate and up-to-date information, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained in this document. Vizrt’s policy is one of continual development, so the content of this document is periodically subject to be modified without notice. These changes will be incorporated in new editions of the publication. Vizrt may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time. Vizrt may have patents or pending patent applications covering subject matters in this document. The furnishing of this document does not give you any license to these patents.

### **Technical Support**

For technical support and the latest news of upgrades, documentation, and related products, visit the Vizrt web site at [www.vizrt.com](http://www.vizrt.com).

### **Created on**

2021/06/22

# Contents

1	Introduction.....	4
2	Context and Architecture .....	5
2.1	Viz Trio Workflow .....	5
2.2	Pilot Edge Workflow .....	6
2.3	DataHub Workflow .....	6
3	Prerequisites.....	7
3.1	Local DCS server .....	7
3.1.1	Hardware.....	7
3.1.2	Software .....	7
3.1.3	Firewall/IPs .....	7
3.2	Viz Social .....	7
4	Installation.....	8
4.1	Viz Social in the Cloud (Performed by Never.no).....	8
4.2	Local DCS Installation (Performed by Vizrt) .....	8
4.2.1	Installation Procedure .....	8
4.2.2	Configure Backup .....	14
4.2.3	Disk Maintenance .....	14
5	Feature Support .....	15
6	Incoming Interface Description .....	16
6.1	Conventions .....	16
6.2	Context .....	16
6.3	Integration.....	16
6.4	Backwards Compatibility .....	16
6.5	Submission of Individual Posts .....	17
6.5.1	Submission via JSON Files .....	17
6.5.2	Submission in Other Formats .....	23
6.6	Submission of Pre-Aggregated Poll Data.....	24
6.6.1	Submission via JSON Files .....	24
6.6.2	Submission in Other Formats .....	27
6.7	Abbreviations .....	27

---

# 1 Introduction

Viz Social is the frontend moderation and content management environment for managing social formats based on social media searches. It is a web application that usually runs as a cloud service in AWS, but if needed can be installed locally on-site as well.

Viz Social feeds curated social content into Vizrt's broadcast environment; where, depending on the architecture, the broadcast playout is controlled by either Viz Trio or by Viz Social itself.

DCS (Dynamic Content Scheduler) is a Windows-based .NET application, functionally similar to MSE. DCS deals with the automation and control of external broadcast devices and graphics engines. It also functions as a scheduler for automation actions and events. DCS is usually installed locally on the broadcaster's network, but it is also possible to install in the cloud: typically when DCS will only be processing.

In this document it's assumed that Viz Social is installed in AWS, while DCS is installed locally in the DMZ.

## 2 Context And Architecture

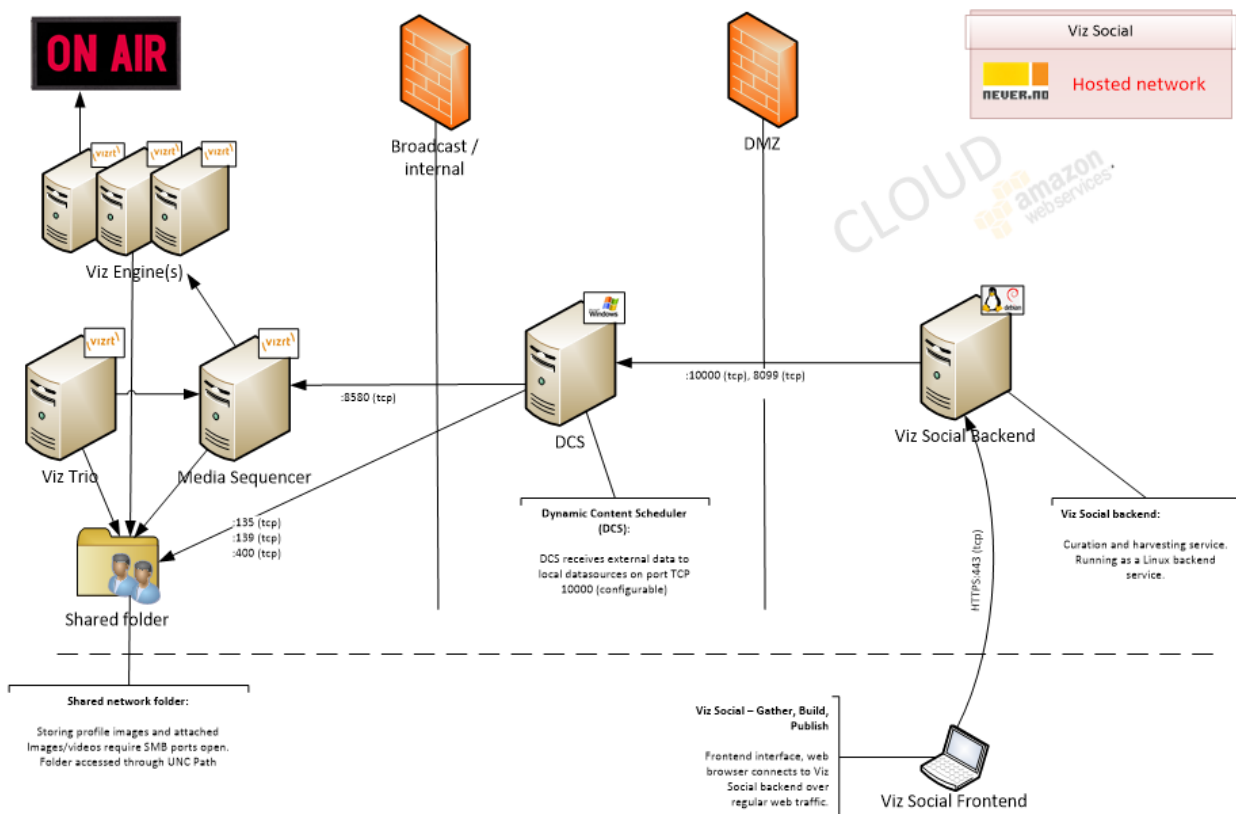
Viz Social runs in Amazon Cloud. The frontend is available on HTTPS, open for specific IPs only (client IPs and support IPs).

DCS is installed locally (usually in a DMZ zone) to be able to serve files and interact with devices and systems that are normally firewalled. To fetch content from the internet (social images/videos), DCS must be able to download/make request outwards.

Viz Social connects to DCS and all communication is done over this channel. It requires opening some firewall ports (default 8099 and 10000) on the DCS for incoming TCP requests from Viz Social in the cloud.

### 2.1 Viz Trio Workflow

DCS talks to the MSE REST API for listing shows and templates, etc. and creates pages and other resources using the same API. In addition, any images/video attachments are saved to a shared folder that Viz Trio clients and the Viz Engines can read from. This is typically either a UNC path, or a mapped network drive that is the same across all machines.

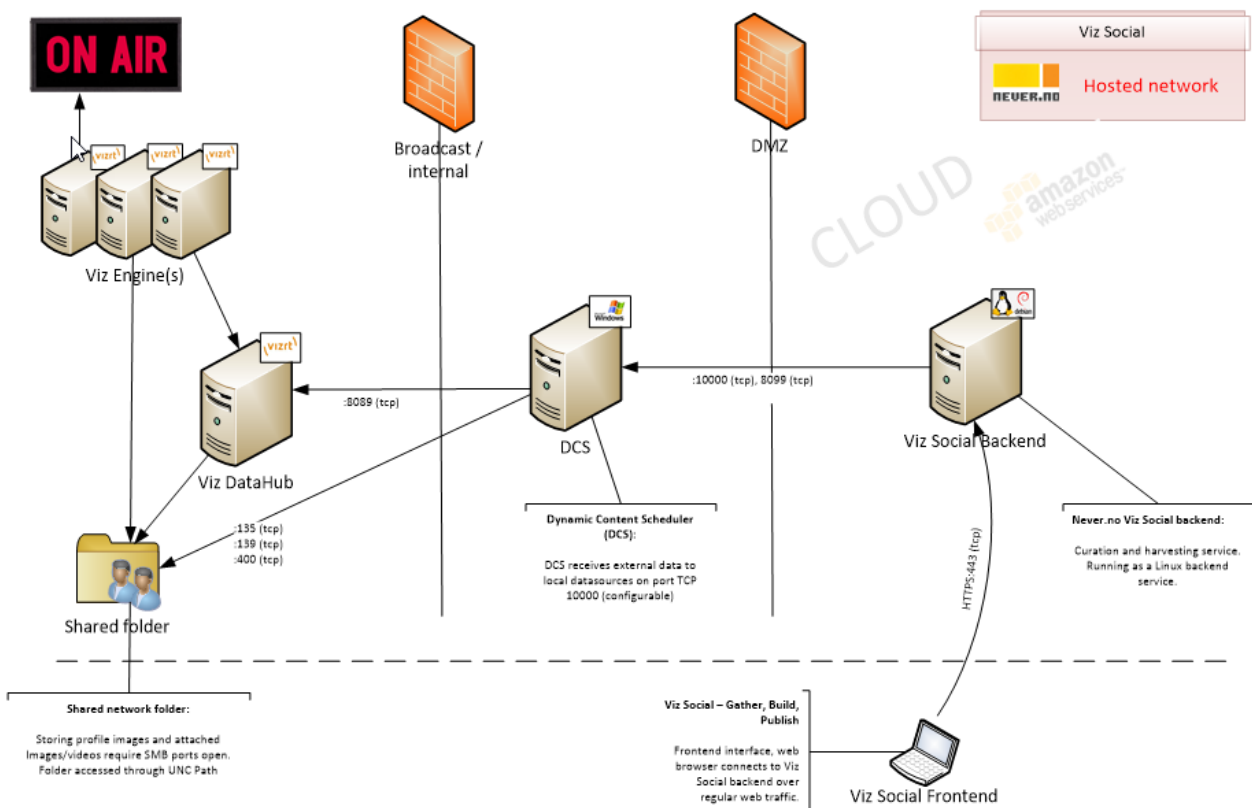


## 2.2 Pilot Edge Workflow

DCS creates ATOM feeds to be used by Pilot Edge.

## 2.3 DataHub Workflow

DCS talks to the Viz DataHub REST API for creating and populating sources. In addition, any images/video attachments are saved to a shared folder that the Viz Engines (and Feed Streamer) can read from. This is typically either a UNC path, or a mapped network drive that is the same across all machines.



---

## 3 Prerequisites

---

### 3.1 Local DCS Server

#### 3.1.1 Hardware

- Desktop with at least a third generation Intel Core i5 Processor.
- Minimum 8GB RAM recommended.

#### 3.1.2 Software

- Windows 2012 Server or newer.
- .NET 4.7.

#### 3.1.3 Firewall/IPs

- A public static IP address for the DCS server.
  - Firewall opened for incoming TCP requests on port 8099 and 10000 from the Viz Social server (IP to open for will be given).
- 

### 3.2 Viz Social

The following information must be provided by the client/Vizrt:

- Client name. Convention for server name is *vizsocial-`<UNIQUECLIENTNAME>.never.no`* (for example, *vizsocial-nrk1.never.no*).
- Location (used to decide which AWS datacenter to install in).
- Fixed public IP addresses the Viz Social interface should be open for.
- The fixed public IP address of the DCS server it will be talking to.
- For Pilot Edge: If ATOM feeds reside in the cloud, the IP address of the Pilot Edge server.

---

## 4 Installation

---

### 4.1 Viz Social In The Cloud (Performed By Never.no)

Viz Social will be installed in the Amazon cloud by Never.no. An AMI and configuration scripts are used for installation. Firewall is opened for IPs provided. One login is given to the client. They can change their password if they want to.

---

### 4.2 Local DCS Installation (Performed By Vizrt)

DCS is installed locally on the client site by Vizrt.

#### 4.2.1 Installation Procedure

Please follow these steps to install DCS:

- Copy over the DCS installer to the machine DCS will run on (*SchedulerInstaller-xxx.exe*).
- Run the installer. By default:
  - DCS is installed in *C:\Program Files (x86)\never.no \DCS 2*
  - Config (*scheduler.xml*) is installed in *%localappdata%\never.no\Dynamic Content Scheduler*  
(e.g. *C:\Users\<USER>\AppData\Local\ never.no\Dynamic Content Scheduler*)
- If DCS is **not** going to run as an administrator user, the REST API for DCS has to be registered with Windows. Open a command prompt in administrator mode (replace red text with correct domain and user):
  - Run: `netsh http add urlacl url=http://+:8099 user=domain\username`
  - If DCS runs as administrator user, this step is not needed.
- Start DCS by clicking the desktop icon. DCS ships with a default configuration file that is used as the basis if none already exists.
- In a browser, go to <http://localhost:8099/index.html> and set up any MSE or DataHub devices (See below).
- If you need to configure any Atom exports:
  - Type `exploreconfig` in the DCS console. This opens up an explorer window with the location where the *scheduler.xml* file is located.
  - Type `persist` in the DCS console window to ensure its saved the configuration file.
  - Close the DCS console window.
- Edit the *scheduler.xml* file in notepad or a similar text editor and apply the customization listed below.
- Save the file, and restart DCS.



## Atom Feeds – Pilot Edge

In the *scheduler.xml* file: find the **xmlexport2** configuration, replace the publishing paths (local and to the cloud if Atom feeds for Pilot Edge are hosted on the Viz Social server in the cloud):

```
<handler name="xmlexport2">
  <entry name="active">True</entry>
  <entry name="error_handling_strategy">Retry</entry>
  <entry name="error_retry_attempts">1</entry>
  <entry name="error_retry_wait_seconds">2</entry>
  <config work_path="%scheduler_configpath%\temp\exp2">

    <datasources>
      <datasource name="social1" source="social1" type="complete" ttl="5" subtype="social" activeparam="active" />
      <datasource name="social2" source="social2" type="complete" ttl="5" subtype="social" activeparam="active" />
      <datasource name="social3" source="social3" type="complete" ttl="5" subtype="social" activeparam="active" />
      <datasource name="social4" source="social4" type="complete" ttl="5" subtype="social" activeparam="active" />
      <datasource name="social5" source="social5" type="complete" ttl="5" subtype="social" activeparam="active" />
    </datasources>

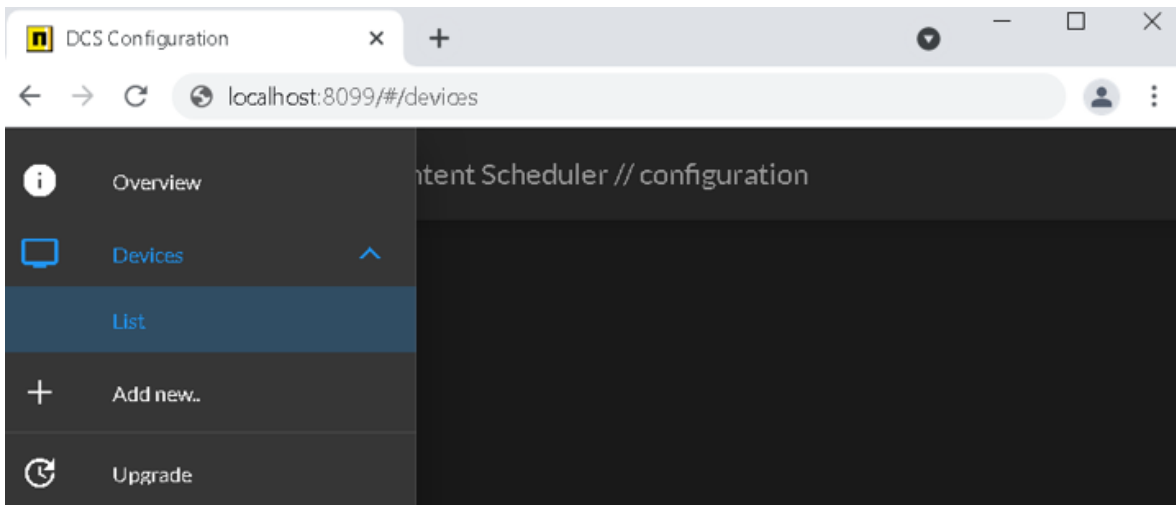
    <target>
      <formatter type="xml" use_new_format="true" filename_prefix="atom" url_prefix="c:\temp\xml\">
        <custom_xslt datasources="**,*" xslt_path="%SCHEDULER_PATH%\XML Export\XSLs\Syndication\Atom feed\VizrtAtomFeed.xsl" indent="true" bom="false"></custom_xslt>
      </formatter>
      <publisher type="localfs" path="c:\temp\xml" />
    </target>
  </config>
</handler>
```

### Note:

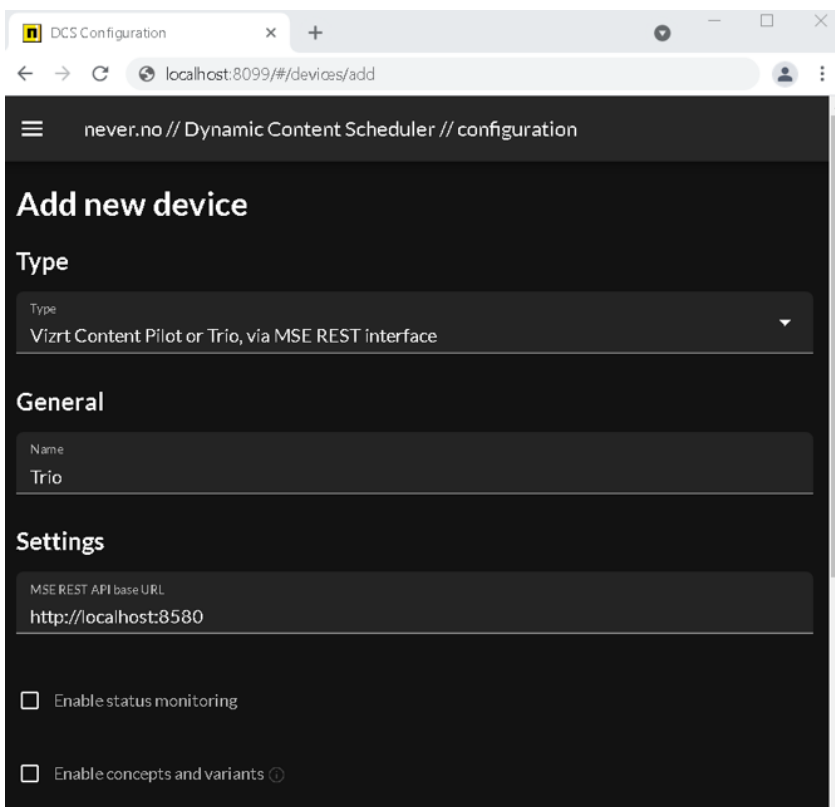
- The *path* attribute of the *<publisher>* element is the actual output location.
- The *url\_prefix* setting on *<formatter>* must end with a slash (This path is rewritten as a *file://* URL in the published Atom (XML) files as the prefix for all *<link>* elements etc.)!

## Viz Trio / MSE

Adding MSE destinations is done using a configuration UI located at <http://localhost:8099/index.html> on the DCS machine.



Click **Add new...**



Give the device a name and choose device type.

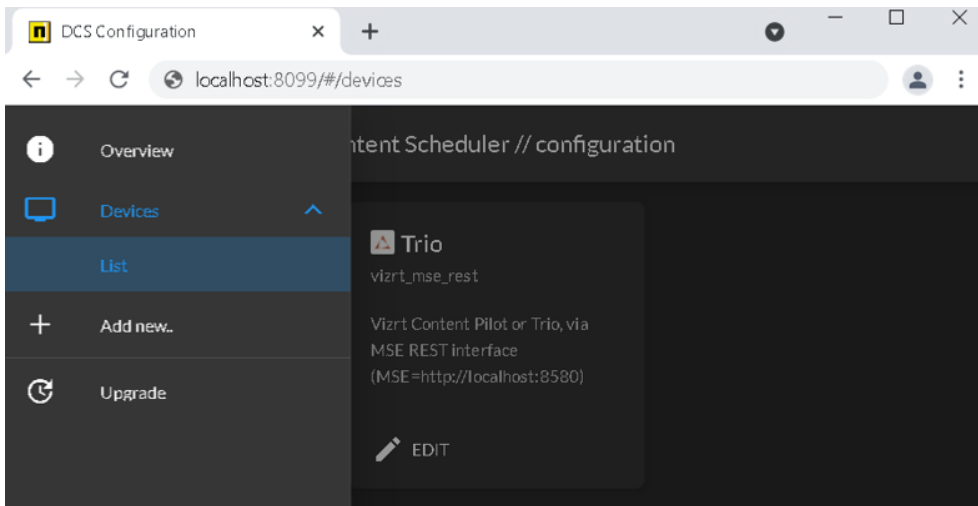
**Device Settings**

MSE REST API Base URL	Base URL of the MSE REST API
Enable status monitoring	Do not enable (work in progress).

MSE REST API Base URL	Base URL of the MSE REST API
Enable concepts and variants	Do not enable (pending missing functionality in MSE REST API).
Enable profile channels	Expose profile channels so that pages can be created on specific, assigned channels (instead of just the default [PROGRAM]).
Enable reverse synchronization of active elements	Monitor active elements in MSE and update cursors in formats (for example, carousels) accordingly (for example, synchronize cue/take actions in Viz Trio back into carousels).
Prioritize using active profiles	Use the currently active profile as selected in Viz Trio instead of using the assigned profile as set up when format was created.
Enable strict template type filtering	Filter out any templates that does not appear to be relevant (does not contain fields relating to social message, poll etc.
Workaround for unescaped backslashes MSE bug	Do not enable unless working with old MSE version (before 5.2).
Disable profile and playout control (export pages only)	Only export pages, do not expose any buttons for controlling playout or profile action.
Attachments path (local)	The local path, relative to DCS machine that attachment files (videos, images) should be saved in.
Attachments path (remote)	What to rewrite the paths as when sending to Viz Trio and Viz Engines etc. Both local and remote path could be <code>\\\\server\share\dir</code> if the UNC path <code>\\server\share\dir</code> is available on DCS, Viz Trio clients, and Viz Engines.
Update attachment timestamps to publishing time (do not preserve originals)	Set the current time when publishing happens, instead of preserving the timestamps as reported by the HTTP Last-Modified header.

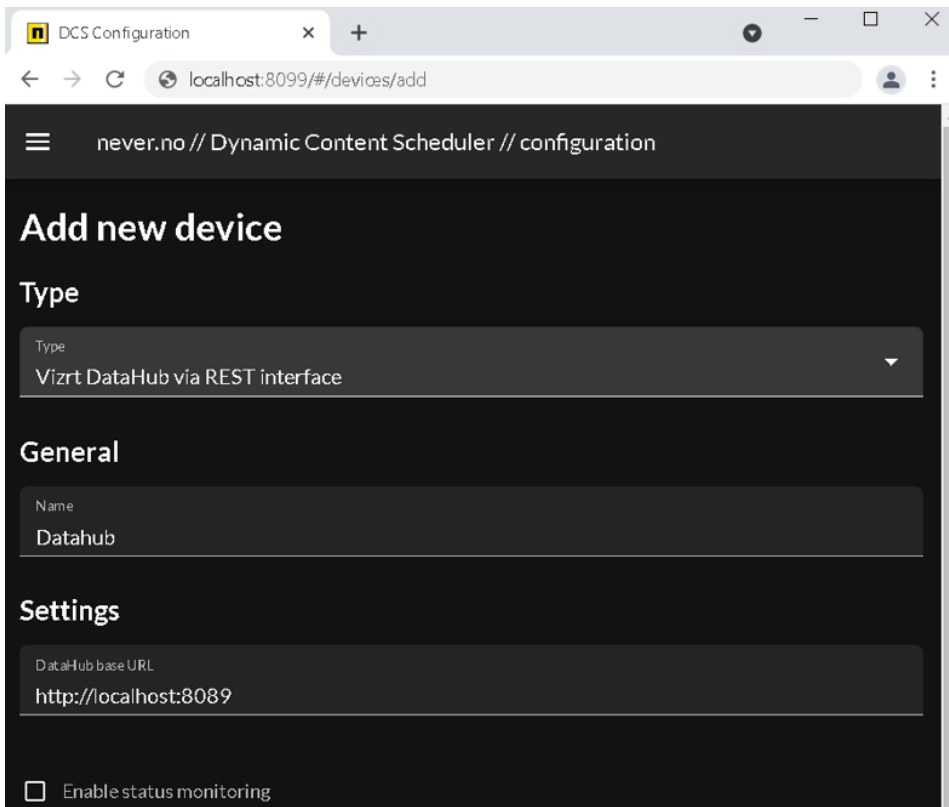
MSE REST API Base URL	Base URL of the MSE REST API
Do not delete attachments when unpublished	Keep attachments on server when unpublishing. <div style="border: 1px solid #ffc107; padding: 5px; margin-top: 10px;"> <b>⚠ Note:</b> The media folder must then be cleaned up manually.                     </div>

After adding you see the device, and you can edit it or add more.




## DataHub / STV

Similar to the Viz Trio configuration UI, configured via <http://localhost:8099/index.html>.



## Device settings

Enable status monitoring	Do not enable (work in progress).
Override images path (local)	Do not override unless you are troubleshooting or have a good reason to set it. By default (and to function properly with the Viz Social plug-ins), the image path as configured in Feed Streamer will be used.
Override images path (remote)	Do not override unless you are troubleshooting or have a good reason to set it. By default (and to function properly with the Viz Social plug-ins), the image path as configured in Feed Streamer will be used.
Update attachment timestamps to publishing time (do not preserve originals)	Set the current time when publishing happens, instead of preserving the timestamps as reported by the HTTP Last-Modified header.

Do not delete attachments when unpublished	Keep attachments on server when unpublishing. <div style="border: 1px solid #f0e68c; padding: 5px; margin-top: 10px;">  <b>Note:</b> The media folder must then be cleaned up manually.         </div>

## 4.2.2 Configure Backup

The *scheduler.xml* file should be backed up to a separate server daily.

## 4.2.3 Disk Maintenance

DCS logs should be deleted after a while. Four weeks of logs should be kept. Logs can be found in *%localappdata%\never.no\Dynamic Content Scheduler*.

Images and Media which have been already downloaded to the DCS local machine reside in a temp folder. These should be deleted after a while. How long to keep images depends on how long after a message is published, the user might take the message On Air.

## 5 Feature Support

This table details the differences in limits between the Premium and Express versions of Viz Social.

Feature	Premium Version	Express Version
Number of accounts	100	1
Simultaneous stories	10+	1
Number of searches	100	15
Twitter API	Firehose API	Standard API
Facebook/Instagram/RSS	Yes	Yes
Youtube	Yes	No
Bespoke and custom feeds (separate quote)	Yes	No
Carousels (simultaneous)	Many	2
Nested Carousels	Yes	
Polls/Competitions	Yes	
Wordcloud	Yes (Coming in 2020)	
Vizrt destinations (Trio, Pilot Edge, MSE, Datahub)	Yes	Yes
Add XML, JSON, HTML5 other Vizrt destinations	Yes	
Customized outputs (separate quote)	Yes	
Moderation tools	Enhanced including reactions	Simple
Chrome extension	Yes	Yes
Analyze and Reports	Yes	
iPad prompter	Yes	

---

## 6 Incoming Interface Description

Viz Social is a collaboration between Vizrt and Never.no for building and producing Social TV formats and Social Advertisements. It is built with the know-how and technology that Never.no has collected by producing online, broadcast and advertising formats for over 15 years. Viz Social offers a standardized framework with an intuitive and easy-to-use interface for all daily operations.

The mission of Viz Social is to make it possible and easy for anyone to produce Social TV and Social Advertisements.

---

### 6.1 Conventions

To differentiate between well-defined concepts used in the Viz Social UI and similar, but not necessarily identical, concepts used in everyday life, the former ones are capitalized in the Viz Social documentation. Examples of these concepts and their spelling are: Carousel, Competition, Filter, Format, Poll, Post, Search, Story, View.

---

### 6.2 Context

Viz Social offers external applications the possibility to share their own data with Viz Social where it can be used for further processing, aggregation, moderation, filtering, scheduling and publishing. Viz Social supports two categories of shared data:

- A series of individual Posts, that is typically inserted and further processed in a Carousel, optionally merged with Posts originating from other sources. These Posts have an originator, a message, a timestamp and can optionally have attachments, an avatar and other attributes.
- A series of snapshots of pre-aggregated data, that are typically processed as the standings of a remotely managed Poll.

Each category is described in a separate section.

---

### 6.3 Integration

In this document, the natively accepted data structures of Viz Social are described. If necessary, Never.no can deploy customizable integration tools for reading feeds of different syntax. Customization options should be discussed with Never.no.

---

### 6.4 Backwards Compatibility

Going forward, Never.no plans that future versions of the interface remain backwards compliant with the current one, assuming.

In parsing data from external sources, the following logic holds:

- No assumption is made about the order of data elements.



- No assumption is made about the presence of optional data elements.
  - Missing mandatory data elements are interpreted as errors.
  - Unrecognized data elements are ignored.
- 

## 6.5 Submission Of Individual Posts

### 6.5.1 Submission via JSON Files

Viz Social is able to accept Posts generated by external sources. Out-of-the-box, this is supported via files formatted in Viz Social's native POST JSON format.

#### Syntax by Example

Each series of external Posts must be formatted as an items array of Posts. Please find an example of such an array below:

```

{
  "items": [
    {
      "id": 1,
      "sticky": true,
      "author": "Małorie Espinoza",
      "nickname": "malorieespinoza",
      "profile_id": 384,
      "profile_image_url": "https://example.never.no/test/img/profile-01.jpg",
      "title": "Message #1",
      "message": "My favourite Pokémon is #291 Ninjask #pokemon #pokemongo ",
      "timestamp": "1534862969020",
      "permalink": "https://example.never.no/messages/index.php?
ref=eyJpZCI6MSwidWlkIjozODQsIm1pZCI6MjkwLCJpaWQiOi0jIsInVybCI6NiwidHMiOi0jE1MzQ4NjI5NjksI
m1zIjoimTUzNDg2Mjk20TAyMCJ9",
      "images": [
        { "url": "https://example.never.no/test/img/pokemon/291.png" },
        { "url": "https://example.never.no/test/img/img-01.jpg" }
      ],
      "videos": [
        { "url": "https://example.never.no/test/video/small.mp4" }
      ],
      "links": [
        { "url": "http://www.pokemon.com/no/pokedex/ninjask" }
      ]
    },
    {
      "id": "2",
      "sticky": false,
      "author": "Iva Carney",
      "nickname": "ivacarney",
      "profile_id": 353,
      "profile_image_url": "https://example.never.no/test/img/profile-02.jpg",
      "title": "Message #2",
      "message": "My favourite Pokémon is #701 Hawlucha #pokemon #pokemongo",
      "timestamp": "1534862969021",
      "permalink": "https://example.never.no/messages/index.php?
ref=eyJpZCI6MiwidWlkIjozNTMsIm1pZCI6NzAwLCJpaWQiOi0jksInVybCI6MSwidHMiOi0jE1MzQ4NjI5NjksI
m1zIjoimTUzNDg2Mjk20TAyMSJ9",
      "images": [
        { "url": "https://example.never.no/test/img/pokemon/701.png" },
        { "url": "https://example.never.no/test/img/img-02.jpg" }
      ],
      "videos": [],
      "links": [
        { "url": "http://www.pokemon.com/no/pokedex/hawlucha" }
      ]
    }
  ]
}

```

```

    "id": "3",
    "sticky": false,
    "author": "Philomena Stephens",
    "nickname": "philomenastephens",
    "profile_id": 685,
    "profile_image_url": "https://example.never.no/test/img/profile-03.jpg",
    "title": "Message #3",
    "message": "My favourite Pokémon is #718 Zygarde #pokemon #pokemongo",
    "timestamp": "1534862969022",
    "permalink": "https://example.never.no/messages/index.php?
ref=eyJpZCI6MywidWlkIjo2ODUsIm1pZCI6NzE3LzJpZWQ0IjYsInVybyCI6NywidHM0jE1MzQ4NjI5NjksI
m1zIjoMTUzNDg2Mjk2OTAyMiJ9",
    "images": [
      { "url": "https://example.never.no/test/img/pokemon/718.png" },
      { "url": "https://example.never.no/test/img/img-03.jpg" }
    ],
    "videos": [],
    "links": [
      { "url": "http://www.pokemon.com/no/pokedex/zygarde" }
    ]
  },

  {
    "id": "4",
    "sticky": false,
    "author": "Beckie Cannon",
    "nickname": "beckiecannon",
    "profile_id": 325,
    "profile_image_url": "https://example.never.no/test/img/profile-04.jpg",
    "title": "Message #4",
    "message": "My favourite Pokémon is #104 Cubone #pokemon #pokemongo",
    "timestamp": "1534862969024",
    "permalink": "https://example.never.no/messages/index.php?
ref=eyJpZCI6NCwidWlkIjozMjUsIm1pZCI6MTAzLzJpZWQ0IjYsInVybyCI6NywidHM0jE1MzQ4NjI5NjksI
m1zIjoMTUzNDg2Mjk2OTAyNCJ9",
    "images": [
      { "url": "https://example.never.no/test/img/pokemon/104.png" },
      { "url": "https://example.never.no/test/img/img-04.jpg" }
    ],
    "videos": [],
    "links": [
      { "url": "http://www.pokemon.com/no/pokedex/cubone" }
    ]
  },

  {
    "id": "5",
    "sticky": false,
    "author": "Janie Goodman",
    "nickname": "janiegoodman",
    "profile_id": 651,
    "profile_image_url": "https://example.never.no/test/img/profile-05.jpg",

```

```

    "title": "Message #5",
    "message": "My favourite Pokémon is #709 Trevenant #pokemon #pokemongo",
    "timestamp": "1534862969030",
    "permalink": "https://example.never.no/messages/index.php?
ref=eyJpZCI6MTAsInVpZCI6NjUxLCJtaWQiOjcwOCwiaWlkIjoxLCJlcmwiOjMsInRzIjoxNTM0ODYyOTY5L
CJtcyI6IjE1MzQ4NjI5NjkwMzAifQ%3D%3D",
    "images": [
      { "url": "https://example.never.no/test/img/pokemon/709.png" },
      { "url": "https://example.never.no/test/img/img-5.jpg" }
    ],
    "videos": [],
    "links": [
      { "url": "http://www.pokemon.com/no/pokedex/trevenant" }
    ]
  }
],
  "paging": {
    "count": 5,
    "min_id": 1,
    "max_id": 5,
    "next_min_id": 6,
    "prev_min_id": null,
    "next_url": "https://example.never.no/messages/index.php?count=5&min_id=6",
    "prev_url": null
  }
}

```

## Semantics

### Post Field Definitions

The following table contains the definitions of the POST fields and parameters.

id	Unique message ID.	Yes
sticky	If set to true, the message is sticky (always shown on top of the posts).	No
author	Full name of the author.	No
nickname	Nickname or alias of the author.	No
profile_id	Profile ID.	No
profile_image_url	URL to profile image / avatar.	No
title	Message title, if applicable.	No
message	Message text.	No
timestamp	Timestamp created. Unix timestamp in milliseconds (other formats can be supported on request).	No
permalink	External link to the posted message.	No
images.url	Array with URLs of attached images.	No
videos.url	Array with URLs of attached videos.	No
links.url	Array with associated links.	No
<b>Field</b>	<b>Description</b>	<b>Required</b>

## Paging

Paging is supported via the dedicated data structure at the end of the example in [Syntax by Example](#). The request from Viz Social uses the request parameters specified in [Request Parameters](#).

## Request Parameters

count	Maximum number of posts per return page.	Yes
min_id	The documents returned should not have an ID lower than <i>min_id</i> .	No
<b>Parameter</b>	<b>Description</b>	<b>Required</b>

## Response Parameters

The response is expected to transmit the response parameters specified by the [Request Parameters](#).

count	Number of posts returned in this page. Might be less than the number that was asked for.	Yes
min_id	Minimal ID returned in this page.	No
max_id	Maximum ID returned in this page.	No
next_min_id	Minimum ID to be used by Viz Social to retrieve the next page.	Yes
next_url	Suggested URL to be used by Viz Social to retrieve the next page.	Yes
<b>Field</b>	<b>Description</b>	<b>Required</b>

**Example:** The response to the query: GET [https://code.never.no/messages/index.php?count=10&min\\_id=11](https://code.never.no/messages/index.php?count=10&min_id=11) might be:

```
{
  "items": [
    {
      "id": 11,
    },
    ..
    {
      "id": 20
    }
  ],
  "paging": {
    "count": 10,
    "min_id": 11,
    "max_id": 20,
    "next_min_id": 21,
    "next_url": "https://code.never.no/messages/index.php?count=10&min_id=21"
  }
}
```

## Business Logic

Please take note of the following business logic when integrating an external source of social Posts:

- The examples in [Syntax by Example](#) show the widest supported set of fields and parameters. Most fields are optional, see [Post Field Definitions](#).
- Post must have a monotonously increasing id.
- The first request from Viz Social usually leaves out *min\_id*. It is up to the server to decide where to set the initial cursor. A suggested option is to respond with the most recent count posts (or less of there aren't that many).
- It could be that no new records are available when a request comes in. In that case, the server is expected to respond with a count of 0, without the *min\_id* and *max\_id*.

## Dynamics

Files with social posts from external sources must be delivered at a well-defined path. File access, transport and naming conventions are not prescribed. After having been agreed to, they form input to Never.no's integration services.

### 6.5.2 Submission in Other Formats

Social posts in alternative JSON formats or even via RSS/Atom feeds or plain XML, can often be supported via customized pre-processing on Viz Social.

**⚠ Important:** Customization options should be discussed with Never.no. Please contact the Never.no support team to discuss options and to come to an optimal solution.

---

## 6.6 Submission Of Pre-Aggregated Poll Data

### 6.6.1 Submission via JSON Files

Viz Social is able to accept aggregated Poll standings generated by external sources. Out-of-the-box, this is supported via files formatted in Viz Social 's native Poll JSON format.

Each snapshot of a pre-aggregated, externally-managed Poll must be formatted as follows.

#### Syntax by Example

Each snapshot of aggregated Poll standings must be formatted as a structure starting with metadata identifying the Poll and its status, followed by counts and other parameters for each Poll alternative. Please find an example of the natively supported structure below:



```
{
  "id": 0,
  "question": "Which is your favourite Pokémon?",
  "image": "https://code.never.no/test/img/profile-01.jpg",
  "status": "open",
  "end_time": 1534864196093,
  "timestamp": 1534777796093,
  "options": [
    {
      "uuid": "a",
      "text": "Pokémon #001: Bulbasaur ",
      "count": 94,
      "image": "https://code.never.no/test/img/pokemon/001.png"
    },
    {
      "uuid": "b",
      "text": "Pokémon #002: Ivysaur ",
      "count": 2,
      "image": "https://code.never.no/test/img/pokemon/002.png"
    },
    {
      "uuid": "c",
      "text": "Pokémon #003: Venusaur ",
      "count": 6,
      "image": "https://code.never.no/test/img/pokemon/003.png"
    }
  ]
}
```

## Semantics

### Poll Field Definitions

The table below contains the definitions of the Poll fields and parameters.

id	An ID used to reference the Poll in question	Yes
question	Question text. Is set to a default text in Viz Social if not included.	No
image	URL to image associated with the question.	No
status	open or closed. Currently ignored by Viz Social.	No
end_time	Unix timestamp (in milliseconds) that the Poll ends.	No
timestamp	Current time of the snapshot as a Unix timestamp (in milliseconds).	No
options	Array of alternatives.	Yes (at least one)
uuid	Unique ID to identify the alternative.	Yes
text	Name of the alternative. Is set to the UUID by Viz Social if not provided.	No
count	Aggregated number of cast votes for the alternative so far.	Yes
image	URL to image associated with the alternative.	No
<b>Field</b>	<b>Description</b>	<b>Required</b>

## Business Logic

Please take note of the following items when designing the business logic for integrating with an external source of Posts:

- The example in [Syntax by Example](#) shows the widest supported set of fields and parameters. Most fields are optional, see [Poll Field Definitions](#).
- Question and alternative texts are only set the first time Viz Social harvests the poll. Updates in later snapshots are ignored, but can be processed (overwritten) manually via the Viz Social UI.
- The read interval to check for updates of the JSON file is configurable on Viz Social.


## Dynamics

Files with aggregated Poll standings from external sources must be delivered at a well-defined path where it can be polled by Viz Social. Only one Poll per JSON file is supported.

File access, data transport and naming conventions are not prescribed. After having been agreed, they form input to Never.no's integration services.

### 6.6.2 Submission in Other Formats

Aggregated Poll standings in alternative JSON formats or even via RSS/Atom feeds or plain XML, can often be supported via customized pre-processing on-board of Viz Social as well.

 **Important:** Customization options should be discussed with Never.no. Please contact the Never.no support team to discuss options and to come to an optimal solution.

## 6.7 Abbreviations

Abbreviation	Meaning
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ISO	International Organisation for Standardization
JSON	JavaScript Object Notation
RSS	Really Simple Syndication
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
XML	eXtended Markup Language