Distributed Caching Provider
User manual

Version 1.1

Table des matières

I. Introduction

Aricie - Distributed Caching Provider (DCP) drives and leverages a caching cloud from a DotNetNuke installation. It allows multiple DNN instances to share cached information throughout the cloud, allowing your server farm to benefit from the cache mechanism globally.

It bridges the gap between DNN's caching API - essentially made for local caches synchronization - and distributed caching technologies - such as Microsoft AppFabric/Velocity or MemCached.

Moreover, Aricie - DCP will organize the flow of cached objects between the nodes of an application farm. Its specific configuration parameters enable to design a caching strategy at multiple levels throughout the application.

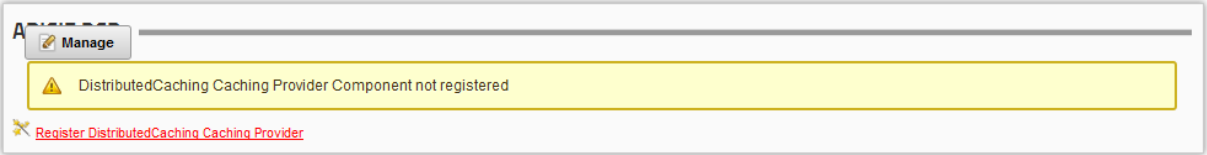
An auto-learning routine makes it easy to initialize the module. This system will log how objects are cached in order to help you understand and tweak the way your application accesses its cache. To speed up the performances of your farm in record time, you can either go GLOBAL – with global optimizations – or LOCAL – with fine-tuned adjustments.

II. Installation

Installing the Aricie – DCP module is handled through the classic DotNetNuke module installation interface. However there are additional steps that you may have to handle by yourself:

- **Working with Velocity or Memcached**? By default only the AppFabric provider is deployed when you install the module. To add the provider you're interested in, just copy the content of the correct folder from <your website path >\DesktopModules\Aricie.DistributedCachingProvider\external\<provider> to <your website path>\bin

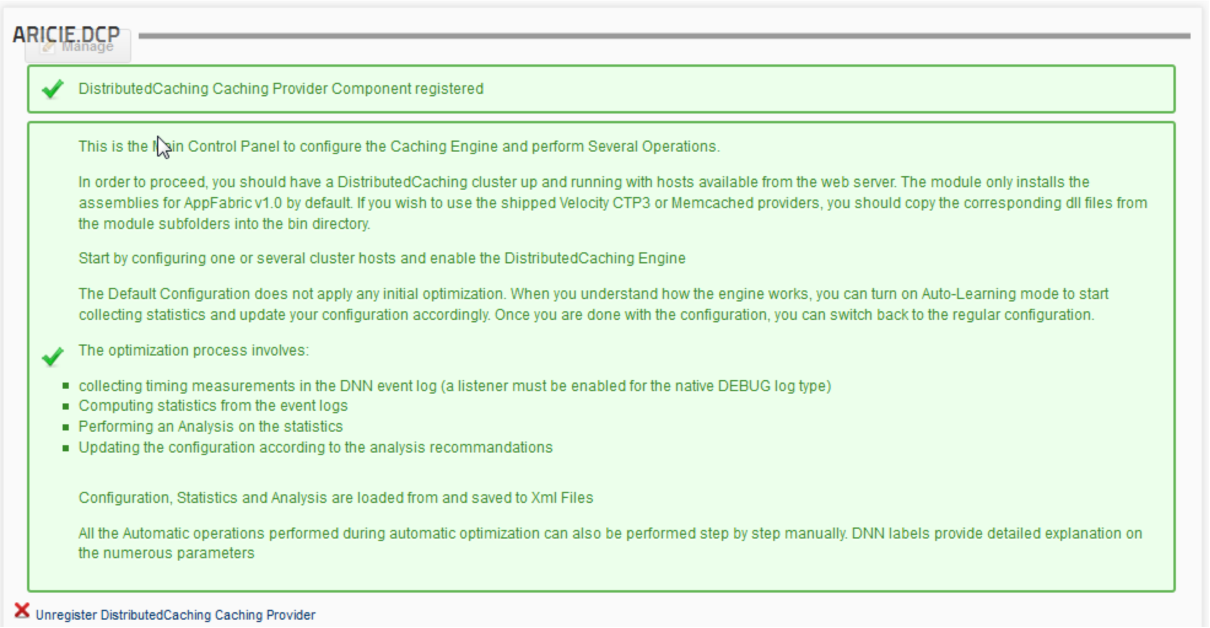
After installing the module, just add it to any page. The module will present its administration interface on this page. First thing you have to do is to register the module to activate it. Just click on the link to install Aricie - DCP:



DistributedCaching Caching Provider Component not registered

Register DistributedCaching Caching Provider

To unregister the module, just click the link that allows you to unregister.



✓ DistributedCaching Caching Provider Component registered

This is the Main Control Panel to configure the Caching Engine and perform Several Operations.

In order to proceed, you should have a DistributedCaching cluster up and running with hosts available from the web server. The module only installs the assemblies for AppFabric v1.0 by default. If you wish to use the shipped Velocity CTP3 or Memcached providers, you should copy the corresponding dll files from the module subfolders into the bin directory.

Start by configuring one or several cluster hosts and enable the DistributedCaching Engine

The Default Configuration does not apply any initial optimization. When you understand how the engine works, you can turn on Auto-Learning mode to start collecting statistics and update your configuration accordingly. Once you are done with the configuration, you can switch back to the regular configuration.

✓ The optimization process involves:

- collecting timing measurements in the DNN event log (a listener must be enabled for the native DEBUG log type)
- Computing statistics from the event logs
- Performing an Analysis on the statistics
- Updating the configuration according to the analysis recommandations

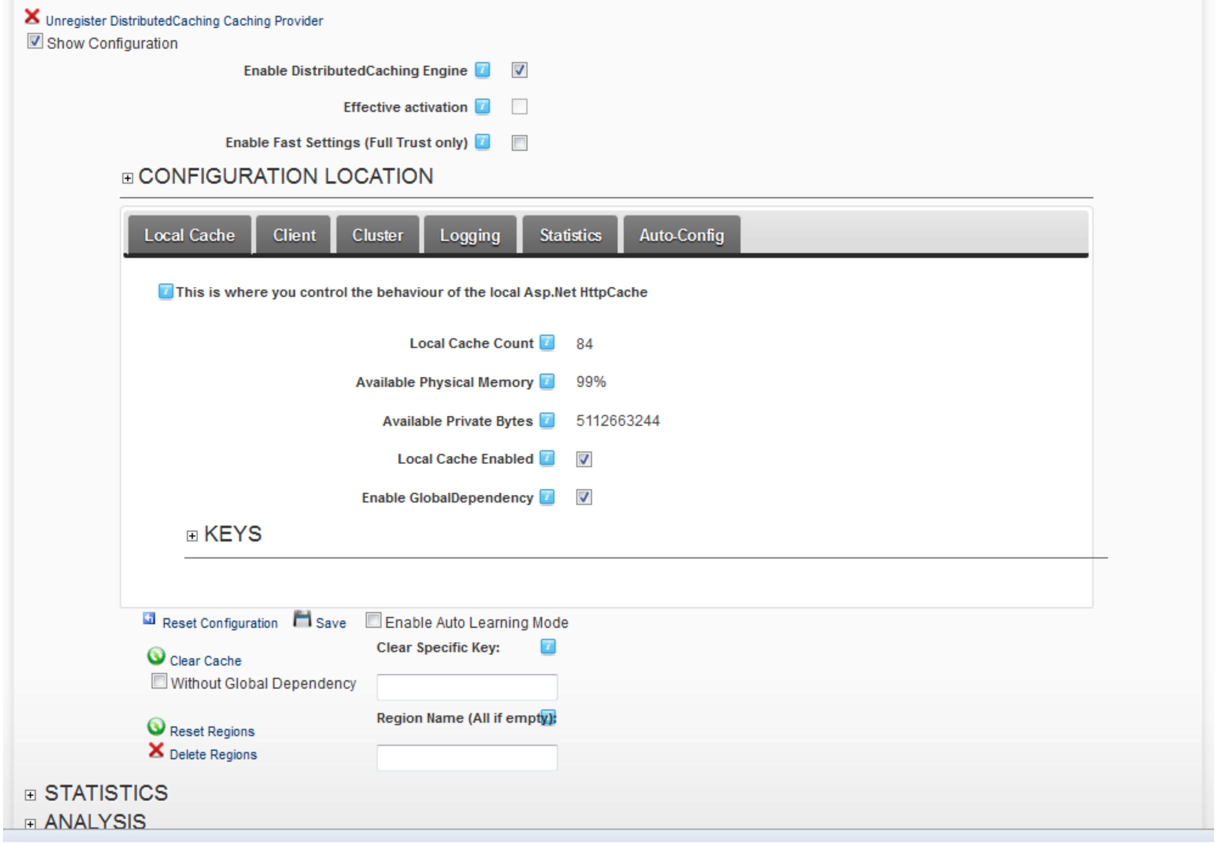
Configuration, Statistics and Analysis are loaded from and saved to Xml Files

All the Automatic operations performed during automatic optimization can also be performed step by step manually. DNN labels provide detailed explanation on the numerous parameters

✗ Unregister DistributedCaching Caching Provider

For now let’s continue and configure the module.

III. Configuration



☑ Show Configuration

Enable DistributedCaching Engine ☑

Effective activation ☐

Enable Fast Settings (Full Trust only) ☐

⊞ CONFIGURATION LOCATION

This is where you control the behaviour of the local Asp.Net HttpCache

Local Cache Count 84

Available Physical Memory 99%

Available Private Bytes 5112663244

Local Cache Enabled ☑

Enable GlobalDependency ☑

⊞ KEYS

☐ Enable Auto Learning Mode

Clear Specific Key:

☐ Without Global Dependency

Region Name (All if empty):

⊞ STATISTICS

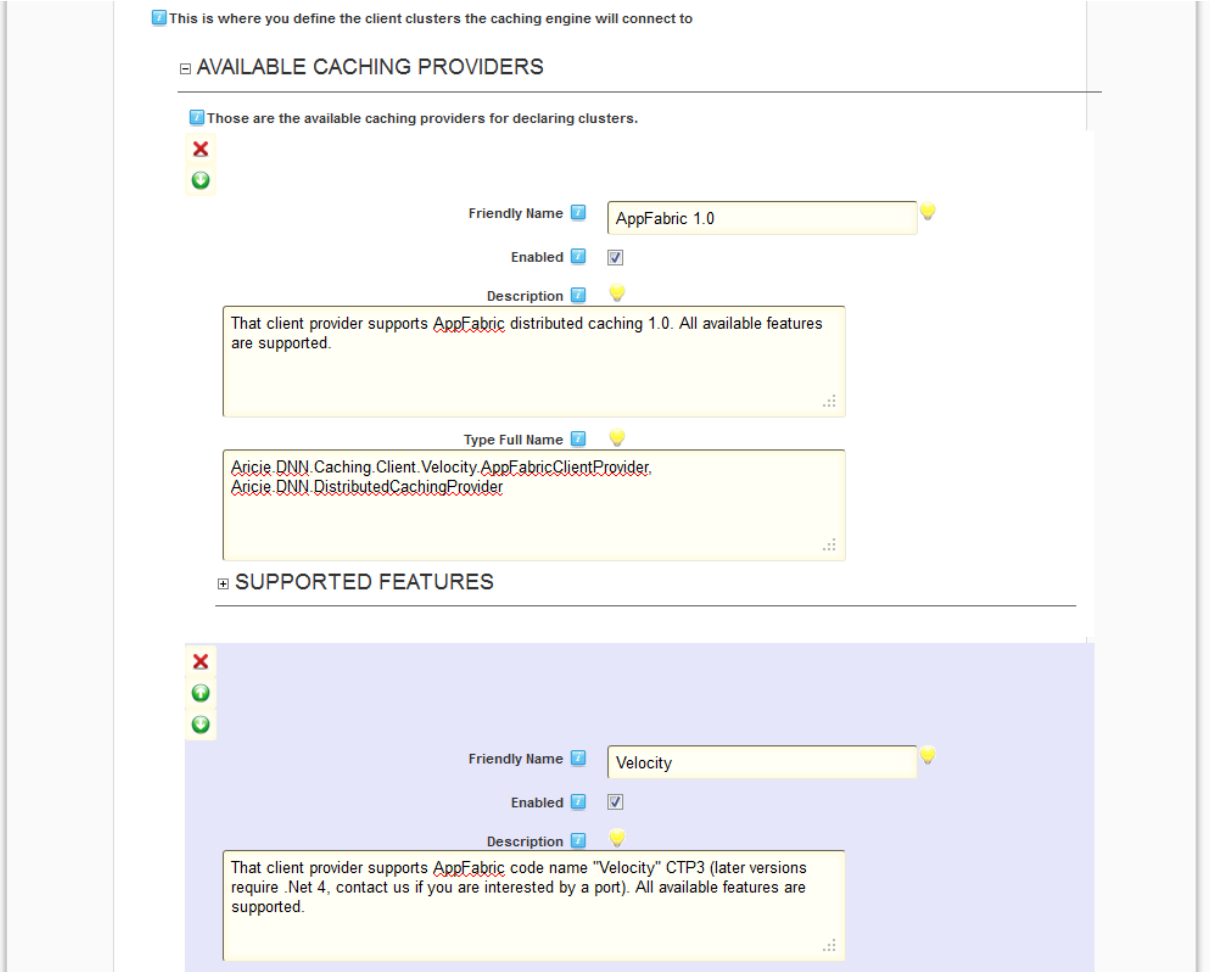
⊞ ANALYSIS

A. Configure the client to access your cached cloud

The first step of your installation should be to configure the distributed system that you'll use for your cloud cache.

Aricie – DCP natively offers drivers for the following cloud cache solutions:

- AppFabric
- Velocity
- Memcache



This is where you define the client clusters the caching engine will connect to

⊟ AVAILABLE CACHING PROVIDERS

Those are the available caching providers for declaring clusters.

Friendly Name: AppFabric 1.0

Enabled: ☑

Description: That client provider supports AppFabric distributed caching 1.0. All available features are supported.

Type Full Name: Aricie.DNN.Caching.Client.Velocity.AppFabricClientProvider, Aricie.DNN.DistributedCachingProvider

⊞ SUPPORTED FEATURES

Friendly Name: Velocity

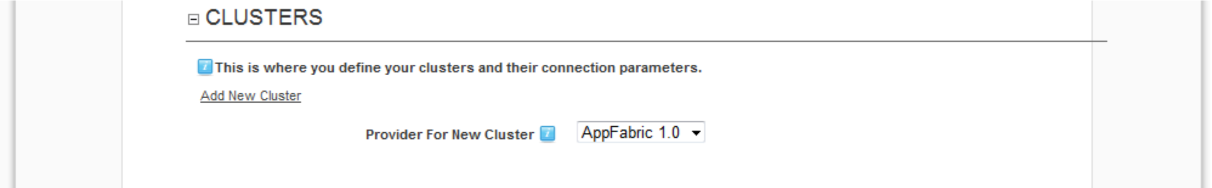
Enabled: ☑

Description: That client provider supports AppFabric code name "Velocity" CTP3 (later versions require .Net 4, contact us if you are interested by a port). All available features are supported.

Even though these systems can differ in their feature sets, the Aricie – DCP drivers can emulate the missing features to level their functionalities and offer a unified configuration interface. Let's add an AppFabric cluster to our configuration in order to connect to this cloud cache.

We'll assume that you've successfully installed and configured an AppFabric instance.

Let's add a new cluster to our client configuration.

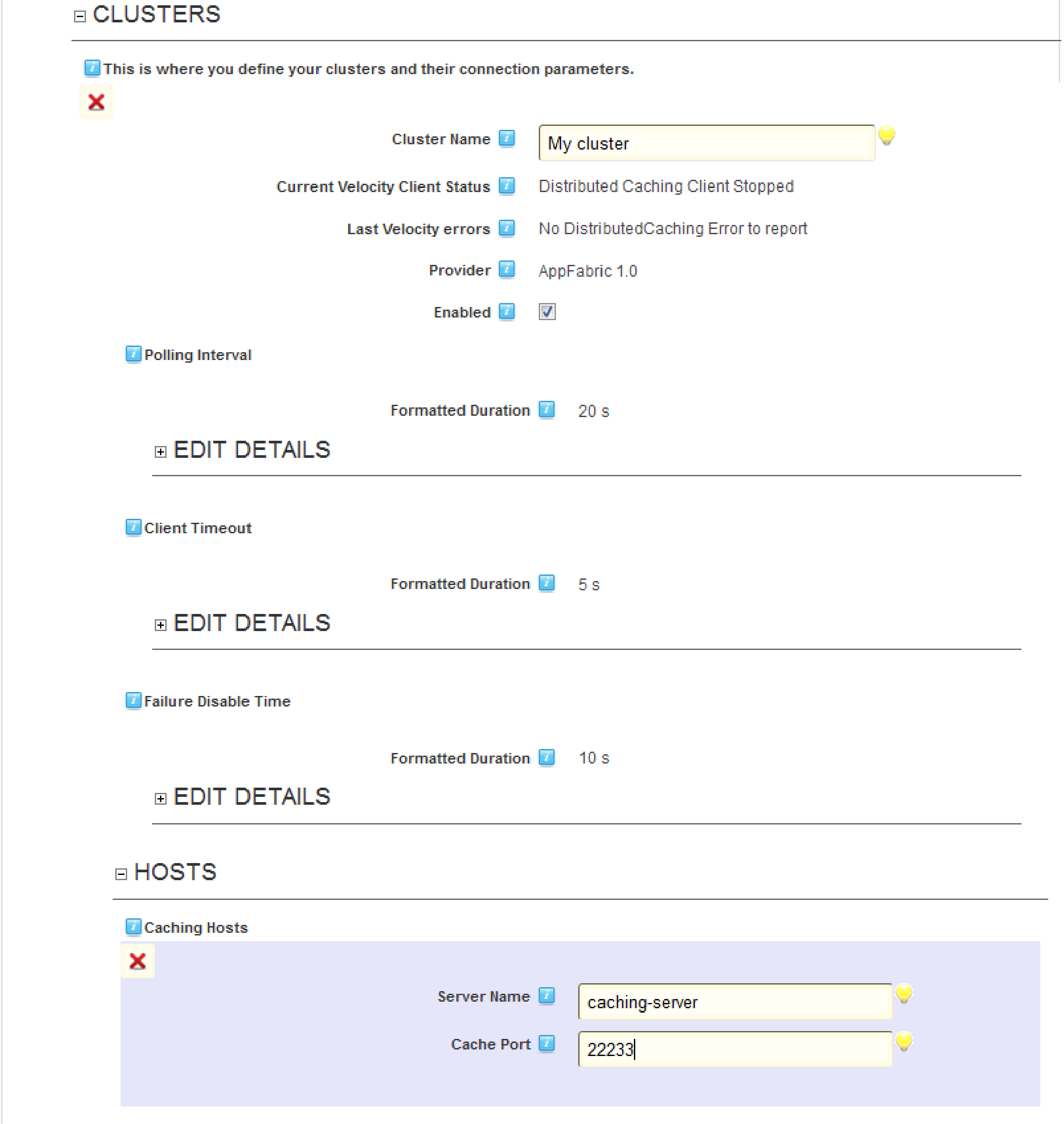


⊟ CLUSTERS

This is where you define your clusters and their connection parameters.

Provider For New Cluster: AppFabric 1.0

After adding the new provider, you'll have to fill in some information regarding the configuration of the client for this cluster.



⊟ CLUSTERS

This is where you define your clusters and their connection parameters.

Cluster Name: My cluster

Current Velocity Client Status: Distributed Caching Client Stopped

Last Velocity errors: No DistributedCaching Error to report

Provider: AppFabric 1.0

Enabled: ☑

Polling Interval

Formatted Duration: 20 s

⊞ EDIT DETAILS

Client Timeout

Formatted Duration: 5 s

⊞ EDIT DETAILS

Failure Disable Time

Formatted Duration: 10 s

⊞ EDIT DETAILS

⊟ HOSTS

Caching Hosts

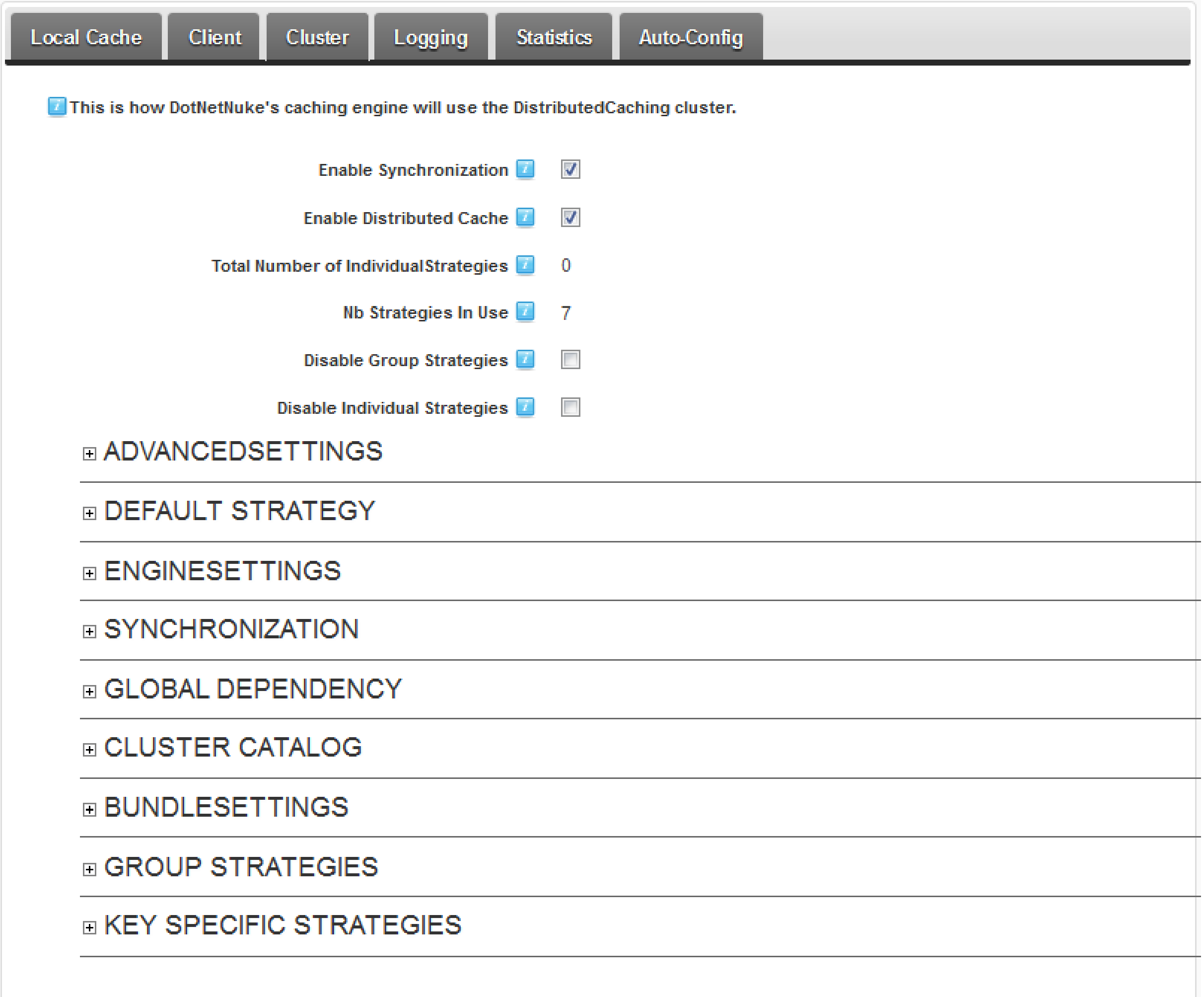
Server Name: caching-server

Cache Port: 22233

After adding the information for your appfabric cluster, your cache is in the cloud you've configured. Navigate your DotNetNuke instance for a bit and check for entries in the cache. (In our example we would for example check the cache statistics with the powershell commandline for AppFabric, but you feel free to use the tool you fancy)

B. Cloud settings

The Cloud Settings configure how DNN interact with the caching clouds.



This is how DotNetNuke's caching engine will use the DistributedCaching cluster.

Enable Synchronization ☑

Enable Distributed Cache ☑

Total Number of IndividualStrategies 0

Nb Strategies In Use 7

Disable Group Strategies ☐

Disable Individual Strategies ☐

⊞ ADVANCEDSETTINGS

⊞ DEFAULT STRATEGY

⊞ ENGINESETTINGS

⊞ SYNCHRONIZATION

⊞ GLOBAL DEPENDENCY

⊞ CLUSTER CATALOG

⊞ BUNDLESETTINGS

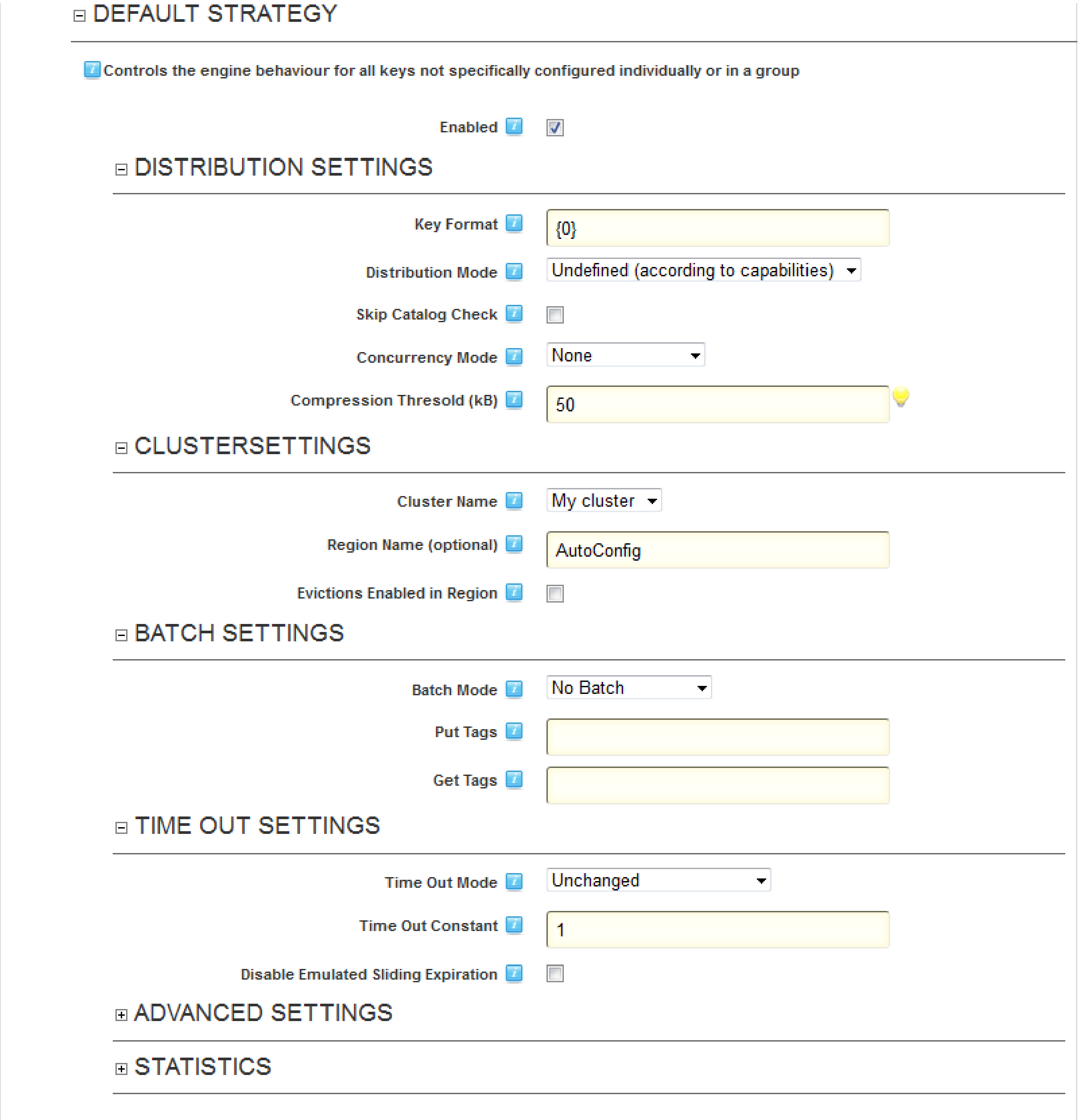
⊞ GROUP STRATEGIES

⊞ KEY SPECIFIC STRATEGIES

C. Caching strategies

DNN cached objects are processed according to what Aricie – DCP calls “caching strategies”, which provide the configuration for the way the caching engine and the target cloud interact.

A default strategy is associated to unknown objects. This default strategy is conservative in order to preserve the system robustness.



⊟ DEFAULT STRATEGY

Controls the engine behaviour for all keys not specifically configured individually or in a group

Enabled ☑

⊟ DISTRIBUTION SETTINGS

Key Format: {0}
Distribution Mode: Undefined (according to capabilities)
Skip Catalog Check ☐
Concurrency Mode: None
Compression Thresold (kB): 50

⊟ CLUSTERSETTINGS

Cluster Name: My cluster
Region Name (optional): AutoConfig
Evictions Enabled in Region ☐

⊟ BATCH SETTINGS

Batch Mode: No Batch
Put Tags:
Get Tags:

⊟ TIME OUT SETTINGS

Time Out Mode: Unchanged
Time Out Constant: 1
Disable Emulated Sliding Expiration ☐

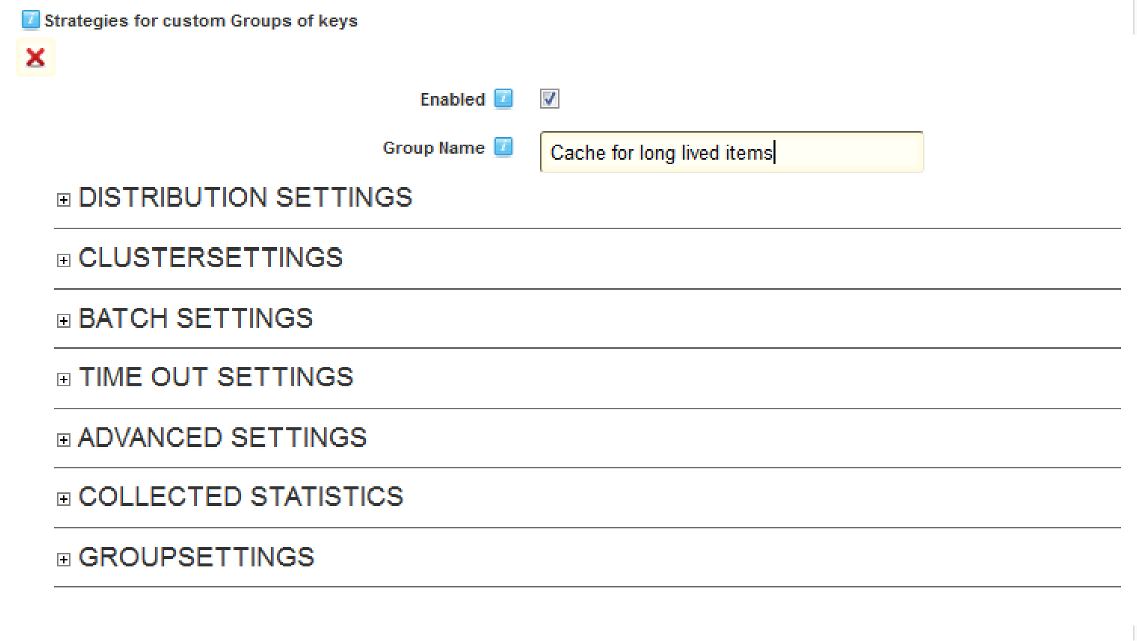
⊞ ADVANCED SETTINGS

⊞ STATISTICS

You can also specify various strategies in order to fine-tune the caching behavior of your application. It's possible for example to configure certain keys in order to keep them from being cached, or cached for a certain duration.

D. Measure and optimize

Of course, the bulk of the configuration is a complicated task, one you can't really take on without good information about the way your application uses the cache provider of DotNetNuke. This information is not easily accessible without knowing the ins and outs of every part of the program you're using. Without this knowledge you cannot evaluate and configure the various strategies optimizing the overall data flow.



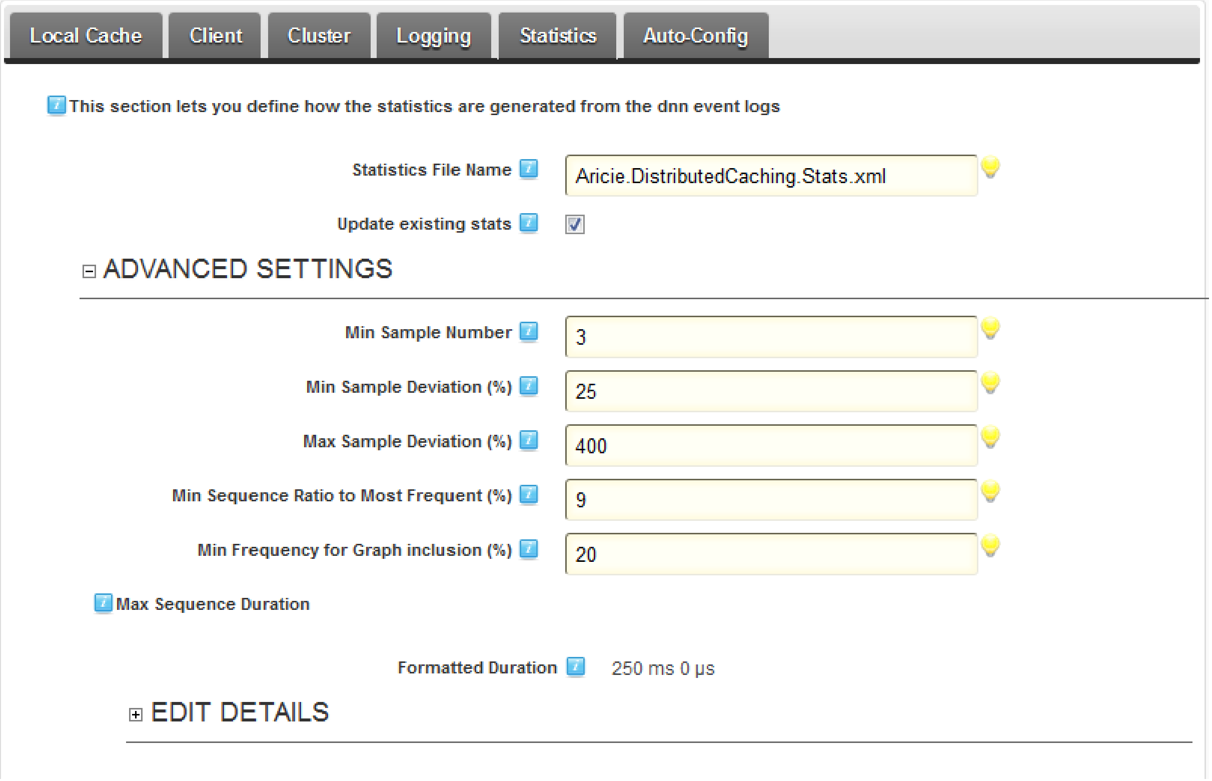
Strategies for custom Groups of keys

Enabled ☑

Group Name: Cache for long lived items

That's where Aricie – DCP Auto Learning tools comes in. The Aricie – DCP comes with a set of tools designed to help you build the optimal strategies based on the usage of your DotNetNuke instance.

The first tool gathers statistics about the real usage of your website in order to build a view of the existing cache topology.



This section lets you define how the statistics are generated from the dnn event logs

Statistics File Name: Aricie.DistributedCaching.Stats.xml

Update existing stats ☑

Min Sample Number: 3

Min Sample Deviation (%): 25

Max Sample Deviation (%): 400

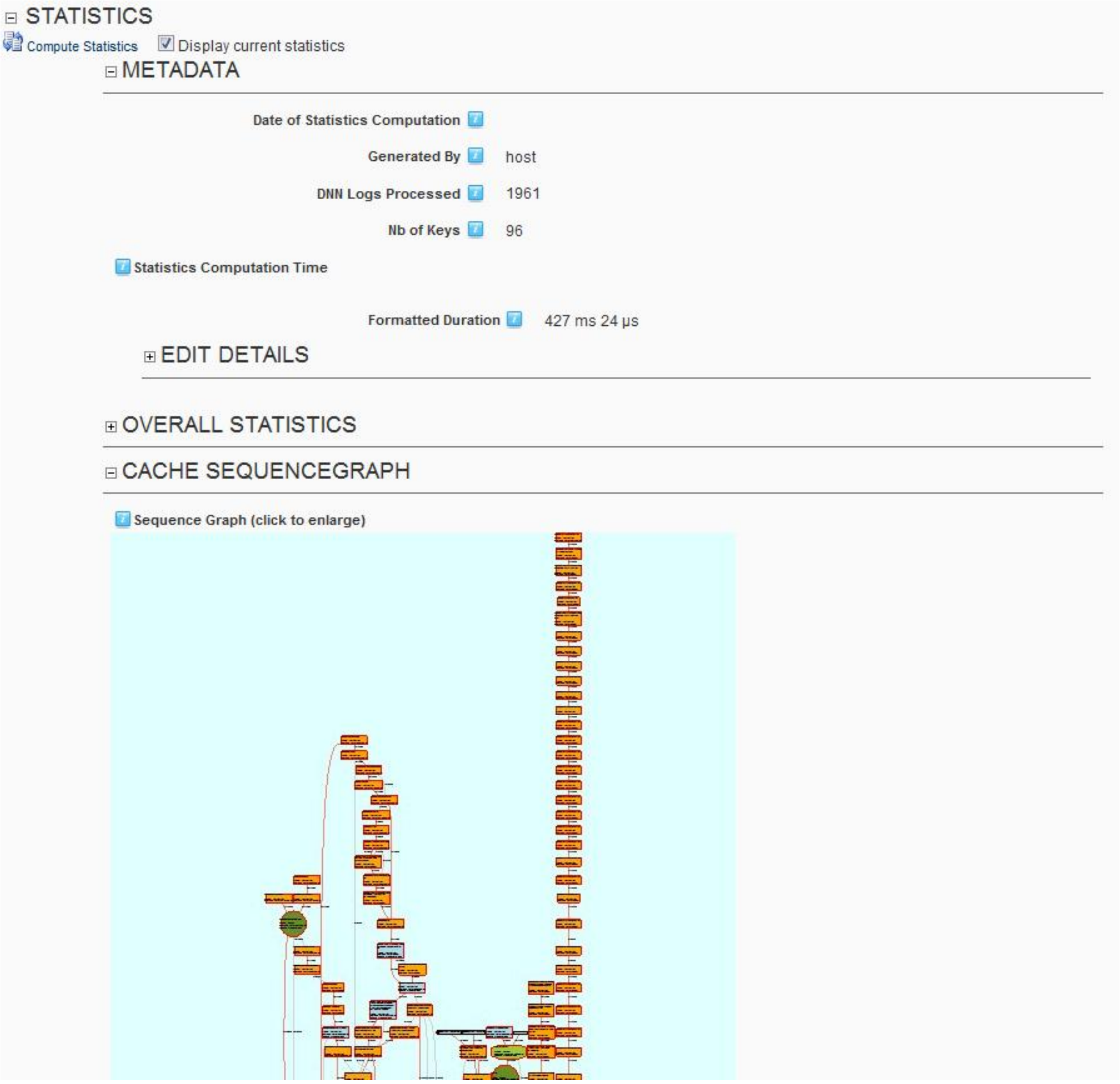
Min Sequence Ratio to Most Frequent (%): 9

Min Frequency for Graph inclusion (%): 20

Max Sequence Duration

Formatted Duration: 250 ms 0 µs

After some measures, the accurate statistics are computed and reported. They provide a broad understanding of how the farm applications behave and leverage the cache.



⊟ STATISTICS

Compute Statistics ☑ Display current statistics

⊟ METADATA

Date of Statistics Computation

Generated By: host

DNN Logs Processed: 1961

Nb of Keys: 96

Statistics Computation Time

Formatted Duration: 427 ms 24 µs

⊞ EDIT DETAILS

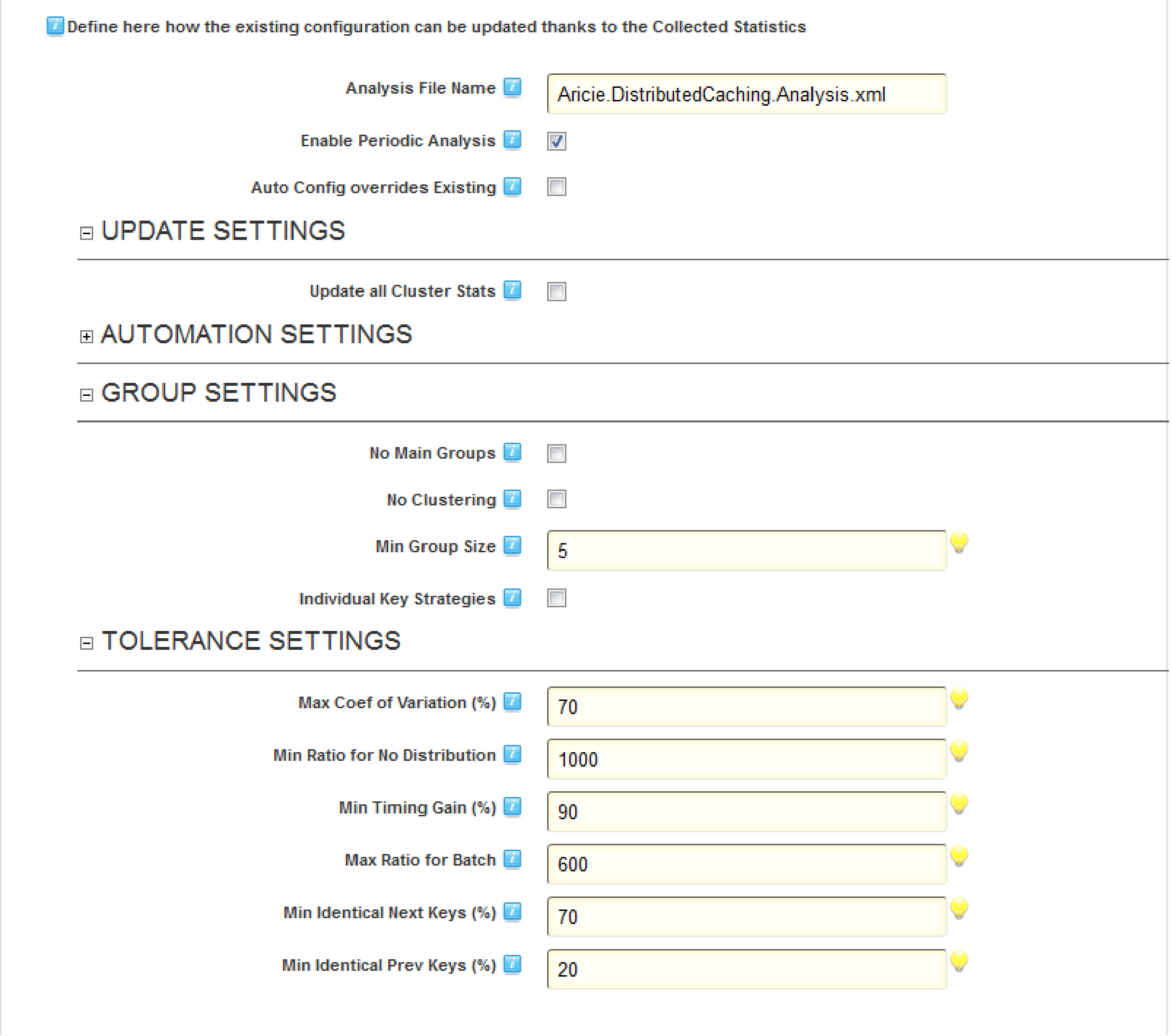
⊞ OVERALL STATISTICS

⊟ CACHE SEQUENCEGRAPH

Sequence Graph (click to enlarge)

A complete training engine can be made after those statistics by the second tool.

With the automation activated, all possible strategies are systematically investigated by the second tool and the corresponding statistics analyzed.



Define here how the existing configuration can be updated thanks to the Collected Statistics

Analysis File Name: Aricie.DistributedCaching.Analysis.xml
Enable Periodic Analysis ☑
Auto Config overrides Existing ☐

⊟ UPDATE SETTINGS

Update all Cluster Stats ☐

⊞ AUTOMATION SETTINGS

⊟ GROUP SETTINGS

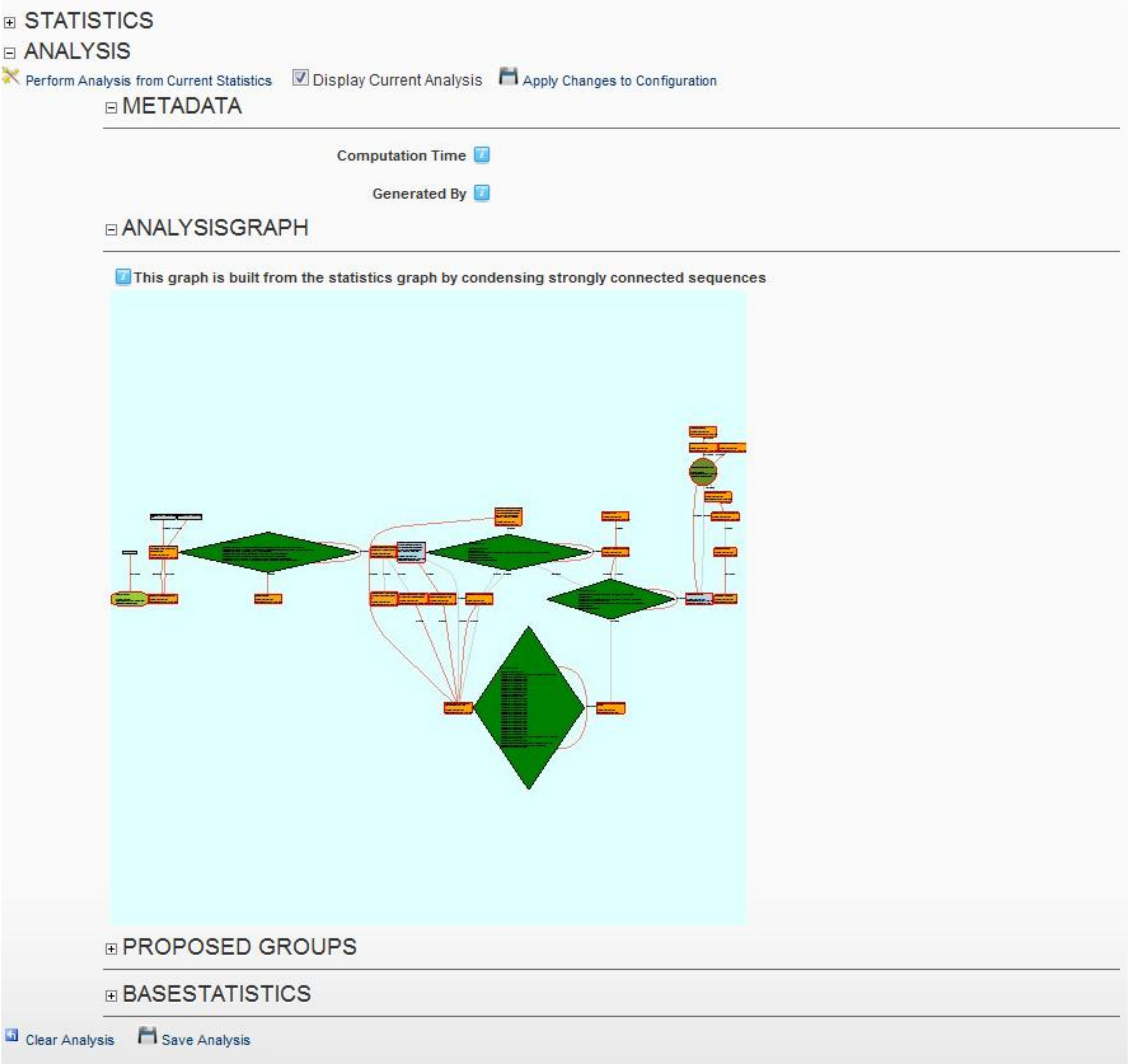
No Main Groups ☐
No Clustering ☐
Min Group Size: 5
Individual Key Strategies ☐

⊟ TOLERANCE SETTINGS

Max Coef of Variation (%): 70
Min Ratio for No Distribution: 1000
Min Timing Gain (%): 90
Max Ratio for Batch: 600
Min Identical Next Keys (%): 70
Min Identical Prev Keys (%): 20

The resulting analysis uses segmentation and graph condensation to propose an optimal set of strategies covering all observed objects.sSuch strategies can be fine-tuned manually before they apply to the overall Settings.

Several complete rounds of analysis can be necessary before the available set of strategies is considered truly optimal. It is important that during the auto-configuration phase, your site usage reflect as much as possible what you’ll expect during the production phase.



⊞ STATISTICS

⊟ ANALYSIS

Perform Analysis from Current Statistics
☑ Display Current Analysis
Apply Changes to Configuration

⊟ METADATA

Computation Time

Generated By

⊟ ANALYSISGRAPH

This graph is built from the statistics graph by condensing strongly connected sequences

⊞ PROPOSED GROUPS

⊞ BASESTATISTICS

Clear Analysis
Save Analysis

IV. Support

We hope you enjoy Distributed Caching Provider and the control it gives you over your caching cloud.

For any question and other suggestions, please contact us on http://www.aricie.com/en/resources/support.aspx