Oracle Endeca Commerce

MDEX Engine Migration Guide

Version 6.4.1 • April 2013

Contents

Copyright and disclaimer

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Preface

The Oracle Endeca Commerce solution enables your company to deliver a personalized, consistent customer buying experience across all channels — online, in-store, mobile, or social. Whenever and wherever customers engage with your business, the Oracle Endeca Commerce solution delivers, analyzes, and targets just the right content to just the right customer to encourage clicks and drive business results.

Oracle Endeca Commerce is the most effective way for your customers to dynamically explore your storefront and find relevant and desired items quickly. An industry-leading faceted search and Guided Navigation solution, Oracle Endeca Commerce enables businesses to help guide and influence customers in each step of their search experience. At the core of Oracle Endeca Commerce is the MDEX Engine™, a hybrid search-analytical database specifically designed for high-performance exploration and discovery. The Endeca Content Acquisition System provides a set of extensible mechanisms to bring both structured data and unstructured content into the MDEX Engine from a variety of source systems. Endeca Assembler dynamically assembles content from any resource and seamlessly combines it with results from the MDEX Engine.

Oracle Endeca Experience Manager is a single, flexible solution that enables you to create, deliver, and manage content-rich, cross-channel customer experiences. It also enables non-technical business users to deliver targeted, user-centric online experiences in a scalable way — creating always-relevant customer interactions that increase conversion rates and accelerate cross-channel sales. Non-technical users can control how, where, when, and what type of content is presented in response to any search, category selection, or facet refinement.

These components — along with additional modules for SEO, Social, and Mobile channel support — make up the core of Oracle Endeca Experience Manager, a customer experience management platform focused on delivering the most relevant, targeted, and optimized experience for every customer, at every step, across all customer touch points.

About this guide

This guide provides instructions to upgrade the MDEX Engine and describes the major changes between versions.

Who should use this guide

This guide is intended for developers who are upgrading Oracle Endeca Commerce, as well as for system administrators managing the product on Windows, UNIX, or Linux.

Note: Unless otherwise indicated, whenever this document specifies UNIX, it applies to Linux as well.

Conventions used in this guide

This guide uses the following typographical conventions:

Code examples, inline references to code elements, file names, and user input are set in `monospace` font. In the case of long lines of code, or when inline monospace text occurs at the end of a line, the following symbol is used to show that the content continues on to the next line: ¬

When copying and pasting such examples, ensure that any occurrences of the symbol and the corresponding line break are deleted and any remaining space is closed up.

Contacting Oracle Support

Oracle Support provides registered users with important information regarding Oracle Endeca software, implementation questions, product and solution help, as well as overall news and updates.

You can contact Oracle Support through Oracle's Support portal, My Oracle Support at *https://support.oracle.com*.

Chapter 1

Upgrading the MDEX Engine

This section describes the steps to upgrade the MDEX Engine. It is important that, after you follow the upgrade procedures in this section, you also review the other sections of this guide for additional changes required to upgrade your specific MDEX Engine implementation.

Recommended reading

In addition to reading this document, Oracle recommends that you read the following documents for important information about the release.

Release Announcement

The Release Announcement provides a brief explanation of the new features that were added in the latest release. The Release Announcement is available as part of the MDEX Engine documentation set that you can download from the Oracle Technology Network.

Release Notes

The Release Notes for each package provide information about new features, changed features, and bug fixes for this release. You can download the latest versions of release notes (`README.txt`) from the Oracle Technology Network.

You can find the Release Notes for core installation packages in:

- The `MDEX/<version>` directory of the MDEX Engine installation.
- The `PlatformServices/<version>` directory of the Endeca Platform Services installation.
- The `CAS/<version>` directory of the Content Acquisition System.
- The `ToolsAndFrameworks/<version>` directory of the Oracle Endeca Tools and Frameworks installation.

Note: While release notes are available with the installation packages, be aware that the latest versions (and possible revisions) of release notes for each package are available on the Oracle Technology Network.

Oracle Endeca Commerce Getting Started Guide

The *Oracle Endeca Commerce Getting Started Guide* gives an overview of Endeca components, describes how to install all the components, and use the Discover Electronics reference application.

You can download the *Oracle Endeca Commerce Getting Started Guide* from the Oracle Technology Network.

Oracle Endeca Commerce Administrator's Guide

The *Oracle Endeca Commerce Administrator's Guide* describes the tasks involved in administering and maintaining applications built with Oracle Endeca Commerce. It bridges the gap between the work performed by the Oracle Endeca Services team when your Endeca implementation is initially deployed, and the issues that your system administrator may need to address to maintain the system.

The guide introduces basic Endeca workflows and environments, and discusses the topology, indicating which physical servers should host specific Endeca components. You can download the *Oracle Endeca Commerce Administrator's Guide* from the Oracle Technology Network.

Package compatibilities

To determine the compatibility of the MDEX Engine with other Endeca installation packages, see the *Oracle Endeca Commerce Compatibility Matrix* available on the Oracle Technology Network.

Upgrading the MDEX Engine from earlier 6.x versions to 6.4.1

This topic describes how to upgrade to MDEX Engine 6.4.1 from the following earlier versions:

- Endeca MDEX Engine 6.3.x
- Endeca MDEX Engine 6.2.x
- Endeca MDEX Engine 6.1.x

To upgrade the MDEX Engine:

1. Stop Endeca services using the **Services** console on Windows, or the `$ENDECA_ROOT/tools/server/bin/shutdown.sh` script on UNIX.
2. Back up the MDEX Engine installation directory of the earlier version.

   This step is useful in case a rollback is necessary. You do not restore any data from this backup after the upgrade.
3. Uninstall the MDEX Engine using the **Add and Remove Programs** utility in the **Control Panel** on Windows, or by using an `rm` command on UNIX.
4. Install MDEX Engine 6.4.1. For instructions, see the *MDEX Engine Installation Guide*.
5. Modify the `PlatformServices/workspace/conf/eac.properties` file, and change `com.endeca.mdexRoot` to the new value of `ENDECA_MDEX_ROOT`.

   (This ensures that the EAC starts MDEX Engine jobs using the correct value for the `ENDECA_MDEX_ROOT` environment variable.)
6. Restart Endeca services by using the **Services** console on Windows, or the `$ENDECA_ROOT/tools/server/bin/startup.sh` script on UNIX.
7. Rerun a baseline update.
8. If you are using the Presentation API to build your front-end application, install it on the server running your Web application. For instructions, see *Presentation API Installation Instructions and Release Notes*.

9. If you are using the Endeca Assembler to build your front-end application, download and install Tools and Frameworks. For instructions, see the *Tools and Frameworks Installation Guide*.
10. If necessary, upgrade other Oracle Endeca Commerce components, such as Endeca Content Acquisition System, Tools and Frameworks, and Endeca Platform Services.

    To determine the compatibility of the MDEX Engine with other Endeca installation packages, see the *Oracle Endeca Commerce Compatibility Matrix* available on the Oracle Technology Network.

After you upgrade

Read the remainder of this guide and complete all required migration changes that may affect your implementation.

Before you add any new features, test your Endeca implementation with the MDEX Engine to make sure that it runs properly with the migration changes you have made. When you are satisfied that your implementation is running as expected, you can start adding any new features that you require.

Chapter 2

Required Changes

You must make the changes specified in this section, if they apply to your application.

Required changes in version 6.4.1

This section contains changes that are required in version 6.4.1.

Required changes in version 6.4.0

This section contains changes that are required in version 6.4.0.

Creating an auxiliary stemming dictionary

In previous releases, you could supplement the Chinese segmentation dictionary with an auxiliary Chinese dictionary. In MDEX Engine 6.4.0, you can supplement the default dictionary for any of the languages that the MDEX Engine supports.

For details about the new configuration procedure, see "About auxiliary dictionaries for OLT analysis" in the *MDEX Engine Advanced Development Guide*.

Presentation APIs are packaged with the MDEX Engine

In MDEX Engine 6.3.0 and later, the Presentation APIs are packaged in the same download as the MDEX Engine.

The Aggregated MDEX Engine (Agraph) is not supported

The Agraph is not supported in the MDEX Engine 6.4.0 release.

For background information, see *The Aggregated MDEX Engine (Agraph) is deprecated* on page 16.

Unsupported Dgraph flags

In MDEX Engine 6.4.0, the `-stat-brel` and `-stat-rel` flags are unsupported and have been removed from the Dgraph.

Deprecated Dgraph flags

In MDEX Engine 6.4.0, the `-d` flag has been deprecated. (The `-d` flag started the Dgraph in debug mode.)

Required changes in version 6.3.0

This section contains changes that are required in version 6.3.0.

The Aggregated MDEX Engine (Agraph) is deprecated

The Agraph is deprecated in the MDEX Engine 6.3.0 release. Existing Agraph implementations will continue to be supported. New implementations should not require the Agraph.

The ability of the Dgraph to store large data sets has improved with each release. These improvements, combined with 64-bit support, have made the Agraph unnecessary.

In the MDEX Engine 6.0 release, the Dgraph added support for 64-bit hardware. Prior to the MDEX Engine 6.0 release, the size of a data set that could be loaded into the Dgraph was limited by the amount of RAM available per process.

Without the process-size restriction, you can load data sets into a Dgraph that would have required an Agraph in releases prior to 6.0. Since that release, the Dgraph has continually improved its scale characteristics to the point that new implementations do not require the Agraph.

Web services and XQuery for Endeca is deprecated

As part of MDEX Engine 6.3.0, the "Web services and XQuery for Endeca" feature is deprecated. This deprecation includes the MDEX Web service (`mdex.xq`), the MDEX API through XQuery (or MAX API), and any custom Web services based on them.

Also, several Dgraph flags that support XQuery are deprecated. This includes `--xquery_fndoc <option>`, `--xquery_path <directory>`, and `--disable_web_services`.

Required changes in version 6.2.2

There are no required changes in version 6.2.2. See the sections on Behavioral Changes.

Required changes in version 6.2.1

This section contains changes that are required in version 6.2.1.

Changes to the silent installation on UNIX

The silent installer on UNIX no longer prompts for acceptance of the License Agreement. In the text file that you create for the silent installation, remove the line that contains "Y" to accept the agreement.

Enablement changes for Endeca Analytics and Record Relationship Navigation

In this release, you no longer enable Endeca Analytics or Record Relationship Navigation using the `ProductConfig.xml` configuration file. Both features are enabled by default. However, each feature requires a separate license.

The *Enabling Record Relationship Navigation* and *Enabling Endeca Analytics* documentation have been removed from MDEX Engine documentation set.

Required changes in version 6.2.0

This section contains changes that are required in version 6.2.0.

Why Did It Match changes

Why Did It Match feature name change

In MDEX 6.2.0, the Why Did It Match feature has been renamed to Why Match to better align with other features such as Why Rank and Why Precedence Rule Fired.

Why Match Dgraph flag deprecation

In MDEX 6.1.4 and earlier, the Why Did It Match feature is enabled by specifying the `--whymatch` flag or the `--whymatchConcise` flag to the Dgraph. In MDEX 6.2.0, the `--whymatch` and `--whymatchConcise` flags are deprecated.

In 6.2.0, the Why Match feature is now enabled on a per query basis using a new `Nx` (Navigation search options) query parameter. The feature in 6.2.0 behaves essentially like Why Match Concise mode in 6.1.4. For details about using this query parameter, see the *Endeca MDEX Engine Advanced Development Guide* and the *Presentation API References* (Javadoc or .NET help).

If you want to continue using the 6.1.4 behavior, you can specify either the `--whymatch` or `--whymatchConcise` flags without implementing the `Nx` parameter in your application. For details about using the 6.1.4 Why Did It Match feature, see the 6.1.4 version of the *Endeca MDEX Engine Advanced Development Guide*. For details about using the 6.2.0 Why Match feature, see the 6.2.0 version of the *Endeca MDEX Engine Advanced Development Guide*.

Why Match API changes

In MDEX 6.1.4 and earlier, the Why Did It Match feature returns information in a property named `Dgraph.WhyDidItMatch`.

In MDEX 6.2.0, the Why Match feature returns a property named `Dgraph.WhyMatch`.

Why Match cross field matches and query expansion

In MDEX 6.1.4 and earlier, the Why Did It Match feature returned a `DGraph.WhyDidItMatch` property that indicated the single field that matched the query terms, the query terms themselves, and information about how the terms matched a field.

In addition to the 6.1.4 information, the `DGraph.WhyMatch` property in MDEX 6.2.0 also includes multiple field names if a cross field match occurred for the query terms, and the `DGraph.WhyMatch` property also includes any query expansion that may have been applied during query processing.

Platform support changes

Added platform support

As of version 6.2.0, the MDEX Engine added supported for the following virtualization environments and operating systems:

- Amazon Elastic Compute Cloud (Amazon EC2)
- VMware vSphere 4.1
- SUSE Enterprise Linux 11

See the *MDEX Engine Installation Guide* for more information.

Removed platform support

As of version 6.2.0, the MDEX Engine removed support for the following platforms:

- Red Hat Enterprise Linux version 4 (both ES and AS)
- Windows Server 2003
- Sun Solaris (all versions)

See the *MDEX Engine Installation Guide* for more information.

The Endeca Control System is not supported

The MDEX Engine 6.2.0 does not support the Endeca Control System. The Endeca Control System includes the Endeca JCD and the Control Interpreter, both of which have been deprecated since Endeca IAP 5.0.

You should use the Endeca Application Controller to control, manage, and monitor components in your Endeca implementation. For details, see *Endeca Platform Services Application Controller Guide*.

The XQuery Data Update API has been removed

The Data Update API which was previously released as an Early Access feature has been removed. This feature will be provided in a future release by an API that incorporates the feedback received during this feature's Early Access program.

Instance configuration files are now stored in the MDEX Engine after Dgidx processing

In MDEX Engine 6.2.0, the instance configuration files (Developer Studio files) are stored in the MDEX Engine after Dgidx processing. This change enables better operational interaction with other Endeca components that rely on the instance configuration files.

The following steps, particularly step 3, illustrate the change to file location:

1. After processing, Forge writes instance configuration files to `<appDir>\data\dgidx_input`. (This behavior has not changed.)
2. Dgidx then loads the instance configuration files from `<appDir>\data\dgidx_input` for processing. (This behavior has not changed.)
3. During processing, Dgidx stores the files to the MDEX Engine (along with the other indexed data) rather than writing the files to `<appDir>\data\dgidx_output`.

The following instance configuration files are affected:

- `<application_name>.analytics_config.xml`
- `<application_name>.derived_props.xml`
- `<application_name>.dimsearch_config.xml`
- `<application_name>.key_props.xml`
- `<application_name>.merchstyles.xml`
- `<application_name>.merchzones.xml`
- `<application_name>.merch_rules.xml`
- `<application_name>.merch_rules_groups.xml`
- `<application_name>.phrases.xml`
- `<application_name>.record_sort_config.xml`
- `<application_name>.recsearch_config.xml`
- `<application_name>.refinement_config.xml`
- `<application_name>.relrank_strategies.xml`
- `<application_name>.render_config.xml`
- `<application_name>.thesaurus.xml`

This change has no migration impact to most applications. However, in rare cases, there may be application impact if an application modified the instance configuration files that Dgidx output before the files were loaded into the MDEX Engine. If this situation applies to your application, you can take the following steps:

1. Copy the instance configuration files from `<appDir>\data\dgidx_input`.
2. Perform any necessary modification on the files.
3. Place the files in the MDEX Engine's input directory.
4. Load the files into the MDEX Engine by running the admin command `config?op=update`.

Flag deprecation, modification, or removal in version 6.2.0

In MDEX Engine 6.2.0, a number of flags have been removed from the Dgraph.

The --aspell flag of the dgwordlist utility has been removed

In version 6.1.3, the `--aspell` flag of the `dgwordlist` utility was deprecated and ignored if specified. In version 6.2.0, the `--aspell` flag has been removed.

For deprecation details, see *Deprecation of the --aspell flag of the dgwordlist utility* on page 43.

The --diacritic-folding flag has been removed from the Dgraph

The `--diacritic-folding` flag to the Dgraph is no longer necessary to match Anglicized search queries such as cafe against result text containing international characters (accented) such as café. You must still specify the `--diacritic-folding` flag to Dgidx to map the international characters to their simple ASCII equivalents.

The `--whymatch` and `--whymatchConcise` flags are deprecated

In MDEX 6.2.0, the `--whymatch` and `--whymatchConcise` flags are deprecated.

If you want to continue using the 6.1.4 behavior, you can specify either the `--whymatch` or `--whymatchConcise` flags without implementing the `Nx` parameter in your application.

Removal of the offline and warmup options from admin?op=update

In the 6.2.0 release, the `offline` and `warmupseconds` options of the `admin?op=update` command are unsupported.

In previous releases, these options were essentially deprecated. You could issue a `admin?op=update` command with the `offline` or the `warmupseconds` options, and the MDEX Engine would ignore the commands, issue a warning, and continue processing the update.

In 6.2.0, support for the `offline` and `warmupseconds` options has been entirely removed. The MDEX Engine no longer issues a warning and stops processing the update if you specify them.

Required changes in version 6.1.5

This section contains changes that are required in version 6.1.5.

Changes to the silent installation on UNIX

The silent installer on UNIX no longer prompts for acceptance of the License Agreement. In the text file that you create for the silent installation, remove the line that contains "Y" to accept the agreement.

Enablement changes for Endeca Analytics and Record Relationship Navigation

In this release, you no longer enable Endeca Analytics or Record Relationship Navigation using the `ProductConfig.xml` configuration file. Both features are enabled by default. However, each feature requires a separate license.

The *Enabling Record Relationship Navigation* and *Enabling Endeca Analytics* documentation have been removed from MDEX Engine documentation set.

Required changes in version 6.1.4

This section contains changes that are required in version 6.1.4.

Presentation APIs are packaged separately from Platform Services

In conjunction with the release of MDEX Engine 6.1.4, Endeca separated the Presentation APIs from Platform Services into separate installation packages.

If you are upgrading to MDEX Engine 6.1.4, you should download, install, and use the Presentation API version 6.1.4. For installation instructions, see the "Installation Instructions and Release Notes for the Presentation API".

Do not use the Presentation API that is available with Platform Services.

The new Presentation API packages are named:

- Presentation API for UNIX which includes the Java version of the API only (JAR files, Javadoc, and Installation and Release Notes file).
- Presentation API for Windows, which includes both the Java version of the API (JAR files, Javadoc, and Installation and Release Notes file) and also the .NET version of the API (DLL files, HTML Help, and Installation and Release Notes file).

The version number (6.1.4) indicates compatibility with the MDEX Engine. For example, the Presentation API version 6.1.4 is compatible with MDEX Engine 6.1.4.

Flag deprecation, modification, or removal in version 6.1.4

The following changes were made to flags that you specify to Forge, Dgidx, or the Dgraph.

Deprecation of the Dgraph --dead_ends flag

Starting with the MDEX Engine version 6.1.4, the `--dead_ends` flag has been removed. If you use this flag, it is ignored by the MDEX Engine

Deprecation of the Dgraph --implicit_exact and --implicit_sample flags

Starting with the MDEX Engine version 6.1.4, the `--implicit_exact` and `--implicit_sample` flags have been removed. If you use these flags, they are ignored by the MDEX Engine.

Deprecation of the --latin1 flag and introduction of the --diacritic-folding flag

In version 6.1.4, the `--latin1` flag to Dgidx and the Dgraph is deprecated and will be unsupported in a future release.

Recall that the `--latin1` flag causes certain international characters in the ISO-Latin1 and Windows CP1252 character sets to be mapped to simple ASCII equivalents. Using this option allows Anglicized search queries such as cafe to match against result text containing international characters such as café.

In version 6.1.4, a new `--diacritic-folding` flag replaces the `--latin1` flag. The `--diacritic-folding` flag is an option to Dgidx and the Dgraph. It is used in the same way `--latin1` was used. However, the `--diacritic-folding` flag supports both Latin1 and additionally supports Latin extended-A.

In the future, `--diacritic-folding` may be expanded to support additional diacritical character folding such as mapping Latin extended-B, extended-C, and so on to their ASCII equivalents. For details, see "Appendix B Diacritical Character Mapping" in the *MDEX Engine Basic Development Guide*.

Required changes in version 6.1.3

There are no required changes in version 6.1.3. Be sure to read about behavioral changes for this version.

Required changes in version 6.1.2

This section contains changes that are required in version 6.1.2.

Removal of all MDEX Engine support for 32-bit processors

32-bit versions of any operating system are not supported by the MDEX Engine in any environment.

Starting with the 6.x release of the MDEX Engine, only 64-bit based hardware and operating systems platforms are supported. Beyond upgrading to 64-bit platforms, no change to the deployment methodology or existing technical artifacts (ITL pipelines, application code, and so on) should be required related to this change. The list of currently supported platforms can be found in the *MDEX Engine Installation Guide*. Any references to 32-bit platforms for software contained within the MDEX installer should be ignored.

Dgraph and Dgidx flag deprecation, modification, or removal in version 6.1.2

In MDEX Engine 6.1.2, the status of the following Dgraph and Dgidx flags has changed. This topic summarizes all the changes.

Changes to the Dgraph flags

The status of the following Dgraph flags has changed:

Flag	Description of change
`-A`	The `-A` flag is deprecated and is not guaranteed to be supported in future releases. The `-A` flag disallows server shutdown and restart through `admin?op=exit` and `admin?op=restart` URL commands sent to the Dgraph.
`--explicit_no_keep_alive`	The `--explicit_no_keep_alive` flag is deprecated and if specified triggers a deprecation warning but is otherwise ignored.

Flag	Description of change
	In previous releases, this flag instructed the MDEX Engine to include a `"Connection: close"` HTTP response header to inform clients that connections are closed after completion of each response. This is now the default behavior.
`--wildcard_approx`	The `--wildcard_approx` Dgraph flag is deprecated and if specified triggers a deprecation warning but is otherwise ignored. The MDEX Engine 6.1.2 uses a different mechanism for wildcard search that does not require using this flag. In previous releases, you could use this flag in some cases to improve performance of wildcard search by allowing approximate wildcard search query matching. For more information, see the topic "Wildcard simplification" in this guide.
`--stat-bins-thresh`	The`--stat-bins-thresh` flag is deprecated and is not guaranteed to be supported in future releases. This flag globally sets the threshold for the maximum number of records above which the MDEX Engine stops computing record counts. By default, the MDEX Engine returns refinement counts for records with no threshold. If you want to speed up processing and do not need the counts, Endeca recommends using the `RECORD_COUNT_DISABLE_THRESHOLD` option in the `refinement_config.xml` file. For more information on this option, see the *Performance Tuning Guide*.
`--stat-bins-cutoff`	The `--stat-bins-cutoff` flag is deprecated and is not guaranteed to be supported in future releases. This flag globally sets the cutoff for record counts. In cases when you do not need to know the exact counts for dimension values (once they are sufficiently high), Endeca recommends using the `MAX_RECORDS_COUNT` option in the `refinement_config.xml` file. For more information on this option, see the *Performance Tuning Guide*.

Changes to the Dgidx flags

The status of the following Dgidx flags has changed:

Flag	Description of change
`--ngram_min`	The `--ngram_min` flag is is deprecated and if specified triggers a deprecation warning but is otherwise ignored.

Flag	Description of change
	In previous releases, you could use this flag for certain types of wildcard search to specify the minimum text substring length to index. For more information, see the topic "Wildcard simplification" in this guide.

Changes to the settings in XML configuration files

The status of settings in some XML configuration files has changed.

Starting with version 6.1.2, the MDEX Engine ignores the following settings and attribute in configuration files and uses a different mechanism for wildcard search:

Setting or attribute	Description and status in 6.1.2
`MAX_NGRAM_LENGTH` `DICTIO¬ NARY_MAX_NGRAM_LENGTH`	Ignored in version 6.1.2. Belong to the `RECSEARCH_INDEXES` and `DIM¬ SEARCH_INDEX` elements in the XML configuration files. In previous releases, these attributes controlled the size of substrings that are indexed for wildcard search.
`DICTIONARY_WILDCARD`	Ignored in version 6.1.2. Belongs to the `RECSEARCH_INDEXES` and `DIM¬ SEARCH_INDEX` elements in the XML configuration files. In previous releases, you had to specify this attribute to enable dictionary-based wildcard search.

Note: Do not remove these attributes or settings from the XML configuration files, since they belong to the DTD used to validate the interface between Dgidx and Developer Studio.

Single assign now enforced on record specifiers

Record files submitted for partial updates must contain exactly one record specifier (also known as a record spec). This condition is now enforced by the MDEX Engine during the Dgraph startup and when running partial updates.

In version 5.1 of the Endeca IAP, the MDEX Engine allowed records to have multiple record specs during startup and when running partial updates.

Starting with version 6.1.2, records can have only one record spec during updates and at startup. Dgidx enforces this restriction when parsing the configuration files for your project. If multiple record spec properties are specified in the `record_spec.xml` file, Dgidx issues an error and the partial update fails.

Before upgrading to this version, ensure that all your records have a single record spec assigned to them.

Two cases are possible in which you can have more than one record spec. Use the following recommendations to eliminate multiple record specs before upgrading:

- If you have duplicate or multiple instances of the same property with the same value, you may have used your record spec property as the join key for a left join in the pipeline. In this case, you should enable the **Remove duplicate property values** checkbox in the record assembler in your pipeline in Developer Studio. This way, multiple copies of the spec property/value will not be assigned to the records.
- If you confirm that the records are coming out of the pipeline with duplicate or multiple record spec values, fix this issue by modifying the Forge pipeline logic and fixing the source data.

Note: As you know from previous releases, a record spec must serve as a unique identifier on all records. For example, if the source data has two different `P_WineID` values on each record, `P_WineID` should not be defined as a record spec.

Required changes in version 6.1.0

This section contains changes that were required in version 6.1.0.

Removal of the http:get-fragment-id() external functions

The XQuery external function `http:get-fragment-id()` has been removed in version 6.1.

Removal of the primary dimension

The primary dimension (also known as the "Endeca" dimension) is no longer required by the MDEX Engine. This change simplifies pipeline creation and records preparation for partial and baseline update processing by the MDEX Engine.

The following sections describe how the primary dimension was used in your project in releases of the Endeca IAP prior to the MDEX Engine 6.1.0, and address the changes that you will see after you upgrade to the MDEX Engine 6.1.0.

To learn how the removal of primary dimension affects your project after an upgrade, select the section that describes your use case.

If you used baseline updates only

Prior to the upgrade to the MDEX Engine 6.1.0, you used baseline updates only. You did not specify the primary dimension explicitly in Developer Studio, and did not tag any records with the primary dimension in your pipelines.

In this case, after an upgrade to 6.1.0, you should do nothing and can still use an `N=0` query for the initial navigation request of your front-end application, as in previous releases.

After an upgrade, Dgidx no longer creates the primary "Endeca" dimension for your project automatically. Forge still creates precedence rules automatically based on your existing dimensions hierarchy. If you issue an initial navigation request of `N=0`, you will continue to see all valid dimensions and refinements being displayed in your front-end application.

This is consistent with the behavior in previous releases with the exception that the "Endeca" dimension is no longer created. In particular, in previous releases, Forge automatically created precedence rules for your project, and the Dgidx automatically created the primary "Endeca" dimension during the baseline update processing. If you issued an `N=0` query to the front-end application, such a query resulted in all valid dimensions and refinements being displayed to end users at an initial navigation request. (Note that the primary dimension that was added to the project automatically did not display, which was an expected behavior for the initial navigation request.)

If you used baseline and partial updates

Prior to the upgrade to the MDEX Engine 6.1.0, you used baseline and partial updates. You also set the primary dimension manually in Developer Studio and tagged all (but not a portion of) records in your project with the primary dimension.

Note: Tagging all records in your project with the primary dimension was recommended in the Endeca IAP version 5.1.x and earlier, as well as in the MDEX Engine 6.0. (This was needed to ensure that the partial updates pipeline could be processed by the MDEX Engine.)

In this case, after an upgrade to 6.1.0, you can still use an `N=0` query as in previous releases. With an `N=0` query, all valid dimensions and refinements will display in the front-end application. Note that the primary dimension that could have been created prior to the upgrade to the MDEX Engine 6.1.0 will never display (this is an expected behavior for the initial navigation request).

If you used the `--unctrct` flag

If you used an `N=id` query along with the `--unctrct` flag for the Dgraph, where `id` is the ID of the previously created primary dimension, do not use the `N=id` query after upgrading to the MDEX Engine 6.1.0. Upon such an `N=id` query, automatic precedence rules will not fire and thus no dimensions and refinements will display. Use an `N=0` query instead to obtain the expected results at the initial navigation state.

If you tagged a portion (not all) of your records with the primary dimension

Read this section only if, prior to upgrading to the MDEX Engine 6.1.0, you tagged only records in partial updates pipeline (but not all records in your project) with the primary dimension.

In this case, after an upgrade to the MDEX Engine 6.1.0, you may end up with some records tagged with the primary dimension and some records not tagged with it. Manually remove the tagging of your records in the partial updates pipeline with the primary dimension, and run the baseline update.

If, after you upgrade to the MDEX Engine 6.1.0, only a portion of your records is tagged with the primary dimension, and you do not remove this tagging, one of two situations is possible: partial updates will be rejected by the MDEX Engine as the primary dimension is not a valid dimension, or the primary dimension will show up as a valid refinement when it should not have been displayed.

Removal of the primary dimension: impact on other features and components

The removal of the primary dimenison in the MDEX Engine 6.1.0 affects other Endeca components. This topic summarizes these changes.

Impact on baseline updates processing

In previous releases, during baseline updates processing, Dgidx had automatically tagged each record with an automatically created primary dimension, also known as the "Endeca" dimension. (The "Endeca" dimension was generated automatically for you by the Dgidx if you did not specify your own primary

dimension explicitly in Developer Studio.) Dgidx also tagged all records in the baseline updates pipeline with this dimension. Starting with version 6.1.0, primary or "Endeca" dimension is no longer created and the tagging does not take place.

If you only used baseline updates, you will not notice any other changes and can still use `N=0` as in previous releases.

Impact on partial updates processing

For partial updates to run successfully, you had to manually tag all records in your partial updates pipeline with a single dimension that you specified in the Developer Studio as the primary dimension. This is no longer necessary. If you tagged all records, you should remove this tagging starting with the MDEX Engine 6.1.0. `N=0` navigation requests continue to work as in previous releases.

Impact on precedence rules

In previous releases, the MDEX Engine handled `N=0` queries using automatically generated precedence rules between the primary dimension and other dimensions. However, starting with MDEX Engine 6.1.0, Dgidx no longer creates the primary dimension during a baseline update if you didn't specify one yourself. This means that precedence rules from the primary dimension to other dimensions are no longer automatically created and do not fire upon `N=0` queries. This should have no impact on you, unless you used `N=id` queries, where `id` is the ID of the primary dimension. If you used `N=id` before the MDEX Engine 6.1.0, use `N=0` after an upgrade.

Developer Studio and removal of primary dimension

The version of Developer Studio that you can use with the MDEX Engine 6.1.0 contains the menu for specifying the dimension as primary. Starting with version 6.1.0, the MDEX Engine ignores changes to the primary dimension performed in Developer Studio and treats all dimensions as secondary.

XML configuration files and removal of primary dimension

In previous releases, the XML configuration files contained the `TYPE` attribute with `SECONDARY | PRIMARY` values. In the MDEX Engine 6.1.0 release, you will notice that the XML configuration files from an upgraded project also contain these values. However, the MDEX Engine 6.1.0 ignores the value `PRIMARY` and treats all dimensions in your project as `SECONDARY` dimensions no matter what is specified in the configuration files.

Note: To open your project in Developer Studio, you must have values specified for the `TYPE` attribute (it should not be empty). After an upgrade, you can either leave your XML configuration files unchanged (in which case the MDEX Engine ignores the `PRIMARY` value for the TYPE attribute of a dimension), or change the `TYPE` values to `SECONDARY` for all dimensions.

The Deployment Template and removal of primary dimension

The Deployment Template project for running partial updates contained a sample record manipulator that could be used to tag all records in the partial updates pipeline with a single primary dimension.

The Deployment Template that works with the MDEX Engine 6.1.0 still contains a sample record manipulator, but you no longer should use it for tagging any records in your partial updates pipeline with a primary dimension since this is not required.

Dgraph flag deprecation, modification, or removal in version 6.1.0

In MDEX Engine 6.1.0, a number of flags have been removed from the Dgraph.

Deprecation of the Dgraph --memusage flag

Starting with the MDEX Engine 6.1.0, the `--memusage` flag has been deprecated. If you use this flag, it is ignored by the MDEX Engine.

In version 5.1.0 of the Endeca IAP, the `--memusage` flag was used to obtain information about the memory usage by the Dgraph data structures.

You can use the MDEX Engine Statistics page to view the MDEX Engine server statistics, including memory usage by the Dgraph.

Deprecation of the Dgraph --ws flag

In MDEX Engine 6.0.1, the `--ws` flag was used to start the Dgraph in Web services mode.

In this release, Web services are enabled in MDEX Engine by default. Therefore, the `--ws` flag has been deprecated and the Dgraph issues a warning message if it is used.

Deprecation of --net_close_timeout behavior

In previous releases, the Dgraph flag `--net-close-timeout` specified the number of seconds the HTTP server would wait after sending a response before forcefully closing down the socket connection with the client.

This timeout interval is important for two reasons:

- Waiting for some time before shutting down the socket ensures that clients get complete responses.
- Timing out after certain period protects against buggy clients, which may never close their end of the socket. This can tie up resources on the server machine, leading to performance degradation and, in the extreme case, denial of service.

In MDEX Engine 6.1, the timeout mechanism is handled differently, so the `–net-close-timeout` flag is no longer necessary and has been deprecated. When the MDEX Engine finishes sending a response to a client, it does a "soft close" of the socket. This allows the client to finish reading data, and to close its end of the socket whenever it is ready. The state of the server-side socket during the interval between the server closing one end, and the client closing the other, is known as `FIN_WAIT_2`. All operating systems supported in this release automatically clean up sockets that stay in `FIN_WAIT_2` for too long.

In general, you should not need to change this from the default value. However, if you want to change the default values, you may do so. For details, see the *MDEX Engine Performance Tuning Guide*.

Removal of the offline and warmup options from admin?op=update

In MDEX Engine 6.1.0, the `offline` and the `warmupseconds` options of the `admin?op=update` command are removed.

In previous releases, you could issue a `admin?op=update&offline=true`, or a `admin?op=update&offine=false` command to the MDEX Engine. Starting with MDEX Engine 6.1.0, the `offline` option of the `admin?op=update` command is removed. If you specify it, the MDEX Engine ignores it. It issues a warning and continues processing the update while keeping the Dgraph port open to incoming queries.

The `warmupseconds` option is also removed. The Dgraph ignores it, issues a warning and continues processing the update.

Removal of --xquery_path default location

The `--xquery_path` flag no longer sets a default location if the location is not specified.

If the flag is not set, a user XQuery path is not used, and user-supplied XQuery modules are not loaded.

Chapter 3

Behavioral Changes

This section describes changes that do not require action on the developer's part, but will have an effect on how an application behaves after you upgrade.

Behavioral changes in version 6.4.1

There are no behavioral changes in version 6.4.1.

Behavioral changes in version 6.4.0

This section lists the behavioral changes in version 6.4.0.

Search results for CJK languages

In MDEX Engine 6.4.0, Chinese, Japanese, and Korean (CJK) data sets are tokenized using a different language analyzer than the analyzer used in 6.3.0. Consequently, search results may vary slightly between 6.3.0 and 6.4.0. You may need to examine your search results to characterize the exact nature of the changes.

Dgraph Statistics page changes

In MDEX Engine 6.4.0, several metrics on the Stats page and the Dgraph request log are calculated differently. The **HTTP: Total request time** and **HTTP: Time reading request** metrics are measured beginning from the first successful socket read from the client, rather than from when the client connection is accepted. The change results in improved accuracy of these reported times.

Behavioral changes in version 6.3.0

There are no behavioral changes in version 6.3.0.

Behavioral changes in version 6.2.2

This section lists the behavioral changes in version 6.2.2.

Request Log Parser packaging and feature changes

In this release, the Request Log Parser is no longer a separate download. It is installed with the MDEX Engine as `$ENDECA_MDEX_ROOT/bin/reqlogparser` (UNIX) and `%ENDECA_MDEX_ROOT%\bin\reqlogparser.exe` (Windows).

Log format requirements

The Request Log Parser no longer supports request logs generated by versions 3.x of the Endeca Information Access Platform. It supports request logs generated by IAP 4.x, IAP 5.x and MDEX Engine 6.x.

Short forms of flag names

In order to be more consistent with other MDEX processes (i.e., Dgidx, Dgraph, Agraph etc), the Request Log Parser no longer supports the short form of flag names. Long form names continue to be supported as in previous releases.

New flag

In order to be more consistent with other MDEX processes , the Request Log Parser now supports a `--version` flag.

Reading from stdin

If the name of a request log is specified as `"-"`, then the request log is read from stdin.

Documentation changes

The *Request Log Parser Usage Guide*, which was distributed a solutions article, has been incorporated into the documentation as "Appendix C Creating Eneperf input files with the Request Log Parser" in the *MDEX Engine Performance Tuning Guide*.

Cheetah naming, packaging, and documentation changes

In this release, the Cheetah utility has been renamed to the Request Log Analyzer, and it is no longer a separate download. It is installed with the MDEX Engine as `$ENDECA_MDEX_ROOT/bin/reqloganalyzer` (UNIX) and `%ENDECA_MDEX_ROOT%\bin\reqloganalyzer.exe` (Windows).

Documentation changes

The *Cheetah Usage Guide*, which was distributed a solutions article, has been incorporated into the documentation as "Appendix E: Using the Request Log Analyzer" in the *Performance Tuning Guide*.

Behavioral changes in version 6.2.1

This section lists the behavioral changes in version 6.2.1.

Enabling Chinese, Japanese, and Korean language support

You no longer need to download and configure separate license keys to enable Chinese, Japanese, or Korean languages. Support for these languages is enabled by default as part of MDEX Engine installation.

Behavioral changes in version 6.2.0

This section lists the behavioral changes in version 6.2.0.

Understanding why a precedence rule fired with the Why Precedence Rule Fired feature

The Why Precedence Rule Fired feature returns information explaining why a precedence rule fired. This information allows an application developer to debug how dimension values are displayed using precedence rules.

For details, see "Using Why Precedence Rule Fired" in the *MDEX Engine Advanced Development Guide*.

Understanding relevance ranking with the Why Rank feature

The Why Rank feature returns information describing which relevance ranking modules were evaluated during a query and how query results were ranked. This information allows an application developer to debug relevance ranking behavior.

For details, see "Using Why Rank" in the *MDEX Engine Advanced Development Guide*.

Dimension search now returns refinement counts

A dimension search can now return refinement counts. To present a more consistent experience to end-users, you can display the refinement counts for dimension search results in the same way you display refinement counts for navigation queries.

You enable refinement counts for dimension search on a per-query basis using the `Drc` (Refinement Configuration for Dimension Search) query parameter.

For details, see "Displaying refinement counts for dimension search" in the *MDEX Engine Basic Development Guide*.

Returning all possible dimension values in a dimension search

You can now create a query that returns all dimension values in all dimensions by specifying `*` (an asterisk) as the string value to the Dimension (`D`) parameter.

For details, see "Returning all possible dimension values in a dimension search" in the *MDEX Engine Basic Development Guide*.

Stemming dictionary changes

Adding a custom stemming dictionary

If your application requires a stemming language that is not available in the Stemming editor of Developer Studio, you can create and add a custom stemming dictionary. A custom stemming dictionary is available in addition to any stemming selections you may have enabled in Developer Studio.

For details, see "Adding a custom stemming dictionary" in the *Endeca MDEX Engine Advanced Development Guide*.

Replacing a default stemming dictionary with a custom stemming dictionary

Rather than supplement a default stemming dictionary, you may chose to entirely replace a default stemming dictionary with a custom a stemming dictionary.

For details, see "Replacing a default stemming dictionary with a custom stemming dictionary" in the *Endeca MDEX Engine Advanced Development Guide*.

Improvements in the Dutch stemming dictionary

The new dictionary provides better search results because it contains improved variants that are based on linguistic declination of proper nouns, nouns, and adjectives. (Prior releases included stemming dictionaries that had more simple rule-based variants.)

For details about stemming, see "Using Stemming and Thesaurus" in the *Endeca MDEX Engine Advanced Development Guide*.

DimensionSearchQuery data type now supports SearchWithinDimensionValueIds

The `SearchWithinDimensionValueIds` element specifies a list of dimension IDs that limit your dimension search to the dimensions you specify. For details see, *Endeca MDEX Engine Web Services and XQuery Developer's Guide*.

Default dimension search can now search against a list of dimension IDs

In version 6.1.4, default dimension search matched a search term against either one dimension or all dimensions. A dimension search against multiple dimensions was not possible without using a workaround that required the Dgidx flag `--compoundDimSearch`.

Furthermore, if you used default dimension search and set the Search Dimension parameter (`Di`) with a list of dimension IDs, the MDEX Engine returned no dimension search results. This behavior has changed in 6.2.0.

In version 6.2.0, you can search against a list of dimensions, rather than just one dimension or all dimensions, by setting the Search Dimension parameter (Di) with a list of dimension IDs. This behavior change applies only to default dimension search. Compound dimension search is unchanged from previous releases.

For additional details, see "Default dimension search", Chapter 16: "Using Dimension Search", and "Di (Search Dimension)" in the *Endeca MDEX Engine Basic Development Guide*.

Dimension value features in partial updates

In this release, you can now use partial updates to add new dimension values that also include dimension value properties (including `Dgraph.Spec` properties). For details, see "Partial update capabilities" in the *MDEX Engine Partial Updates Guide*.

New considerFieldRanks parameter for relevance ranking modules

The relevance ranking modules Exact, First, Freq, Nterms, Numfields, Phrase, and Proximity can now accept an optional query-level parameter named `considerFieldRanks`.

If specified as part of a query, the `considerFieldRanks` parameter indicates that the module further sort records according to field ranking scores, in addition to sorting according to the standard behavior of the module. For details about usage with each relevance ranking module, see "Controlling relevance ranking at the query level" in the *MDEX Engine Advanced Development Guide*.

Note: The `considerFieldRanks` parameter is a query level parameter that overrides the search interface settings you configure in Developer Studio. The capability to consider field ranks cannot be specified in the Relevance Ranking Modules editor of Developer Studio.

Behavioral changes in version 6.1.5

This section lists the behavioral changes in version 6.1.5.

Enabling Chinese, Japanese, and Korean language support

You no longer need to download and configure separate license keys to enable Chinese, Japanese, or Korean languages. Support for these languages is enabled by default as part of MDEX Engine installation.

Behavioral changes in version 6.1.4

This section lists the behavioral changes in version 6.1.4.

New or revised flags in version 6.1.4

The following changes were made to flags that you specify to Forge, Dgidx, or the Dgraph.

The --stemming-updates flag for Dgidx is new

You can supplement the default stemming dictionaries by specifying a flag to Dgidx (--stemming-updates) and providing an XML file of custom stemming changes. The stemming update file may include additions, deletions, or combinations of both. For usage details, see the MDEX Engine Advanced Development Guide.

The --failedupdatedir flag for the Dgraph is new

You can use the new `--failedupdatedir <dir>` flag for the Dgraph to specify the directory in which the MDEX Engine should store the failed update files.

The default directory that the MDEX Engine uses for storing the failed update files is `<updatedir>/failed_updates/`.

The --phrase_max flag for the Dgraph is new

You can use the new `--phrase_max <num>` flag for the Dgraph to specify the maximum number of words in each phrase for text search. Using this flag improves performance of text search with phrases.

The default number is 10. If the maximum number of words in a phrase is exceeded, the phrase is truncated to the maximum word count and a warning is logged.

Support for VMware ESX 4

The MDEX Engine supports VMware ESX 4 for the following guest operating system platforms:

Platform	Recommended MDEX Engine server configuration
RHEL 5	• Configure four VCPUs on a virtual machine • Allocate a single Dgraph per virtual machine • Specify four threads for each Dgraph
Windows 2008 R2	• Configure four VCPUs on a virtual machine • Allocate a single Dgraph per virtual machine • Specify four threads for each Dgraph

The number of threads should not exceed the number of VCPUs. Endeca does not recommend running more than one MDEX Engine per virtual machine.

Note: VMware configurations of the MDEX Engine are not supported with the Agraph.

For additional information on VMware support and performance, see the *Performance Tuning Guide*.

MDEX Engine cache improvements

The MDEX Engine cache has been improved to allow it better handle performance requirements of Endeca applications with a high number of threads.

Three new columns have been added to the Cache tab in the MDEX Engine Stats page. The data in these columns can be useful to you if you need to analyze and tune the cache or re-design the front-end application to improve performance of the MDEX Engine.

- These new columns are added to the Cache tab of the MDEX Engine Stats page:
  - “Number of rejections”. Examining this column is useful if you want to see whether you need to increase the amount of memory used for the MDEX cache. Counts greater than zero in this column indicate that the cache is undersized and you may want to increase it.
  - “Number of reinsertions”. Examining this column is useful if you want to examine your queries for similarities and improve performance by considering the redesign of the front-end application. Large counts in the "Number of reinsertions" column indicate that simultaneous queries are computing the same values, and it may be possible to improve performance by sequencing queries, if the application design permits.
  - "Total reinsertion time". Examining this column is useful for quantifying the overall performance impact of queries that contribute to the "Number of reinsertions" column. This column represents the aggregated time that has been spent calculating identical results in parallel with other queries. This is the amount of compute time that potentially can be saved by sequencing queries in a re-design of the front-end application.
- The “Live references” column is removed from the Cache tab.

Cache settings and performance tuning

As a result of the cache changes in version 6.1.4, the cache settings you used in 6.1.3 may or may not be the most appropriate settings in 6.1.4. After upgrading to 6.1.4, you should tune the cache setting `--cmem` and check the resulting performance to ensure the Dgraph cache performs as expected.

For guidance, see "Cache-tuning recommendations for optimizing performance" in the *Endeca MDEX Engine Performance Tuning Guide*.

Merge policy for partial updates

The MDEX Engine now incorporates a merge policy that specifies how frequently it merges partial update generations.

Generation files are combined through a process called *merging*. Merging is a background task that does not affect MDEX Engine functionality but may affect performance. Because of this, you can set the policy that dictates the aggressiveness of the merges; this policy is called the *merge policy*.

You can control the merge policy through a Dgraph flag or through the admin interface via a URL command. Using these controls, you can set the merge policy to one of two settings:

- A `balanced` policy that strikes a balance between low latency and high throughput. This is the default policy of the MDEX Engine.
- An `aggressive` policy that merges frequently and completely to keep query latency low at the expense of average throughput.

The Dgraph `--mergepolicy` flag can be used to set the merge policy at start-up time. If this flag is not used, the merge policy defaults to the `balanced` policy, which is comparable to how versions 6.1.0 through 6.1.3 of the MDEX Engine handle merges.

The new URL `mergepolicy` command can be used to force a merge, and (optionally) to change the merge policy of a running MDEX Engine.

See the *Partial Updates Guide* for details on the merge policy, as well as on the Dgraph `--mergepol¬icy` flag and the URL `mergepolicy` command.

Displaying disabled refinements

Disabled refinements represent those refinements that front-end application users could reach if they were to remove some of the top-level filters that have been already selected from their current navigation state.

In many front-end applications, it is desirable to have a user interface that allows users to see the impact of their refinement selections. In particular, once the users make their initial selections of dimensions and refine by one or more of them, it is often useful to see not only the refinements that are available at each step in the navigation but also the disabled refinements that would have been available if some of the other selections were made.

Such refinements are typically displayed in the front-end application as grayed out, that is, they are not valid for clicking in the current state but could be valid if the navigation state were to change.

To configure disabled refinements, you do not need to change the Endeca project configuration XML files used with Forge, Endeca Workbench, and Developer Studio. You also do not change any settings in the Endeca Workbench and Developer Studio. No changes are required to existing Forge pipelines. The index format of the Dgidx output does not change.

Front-end application developers who wish to display disabled refinements need to introduce specific front-end application code that augments queries with the configuration for disabled refinements.

You configure the display of the disabled refinements on a per query basis. You can do this using either of these approaches:

- Presentation API methods, or URL parameters. For information, see the section "Displaying disabled refinements" in the *MDEX Engine Basic Developer's Guide*.
- The MDEX XQuery (MAX) API (if you are using XQuery and Web services for Endeca). For information, see the *XQuery and Web Services Developer's Guide*.

Retrieving refinement counts for records that match descriptors

For each dimension that has been enabled to return refinement counts, the MDEX Engine returns refinement counts for records that match descriptors. Descriptors are selected dimension values in this navigation state.

The refinement counts that the Dgraph returns for descriptors are returned with the `DGraph.Bins` or `DGraph.AggrBins` property on the descriptor `DimVal` object returned through the Endeca navigation API.

The count represents the number of records (or aggregate records, in the case of `DGraph.AggrBins`) that match this dimension value in the current navigation state.

This capability of retrieving refinement counts for descriptors is the default behavior of the MDEX Engine. No additional configuration (for example, Dgraph command line options) is needed to enable this capability.

For detailed information on retrieving counts for records that match descriptors, see the topic in the *MDEX Engine Basic Developer's Guide*.

If you are using XQuery for Web Services at Endeca and need information on retrieving refinement counts for descriptors, see the *MDEX Engine XQuery and Web Services Developer's Guide*.

The DVAL_STATIC_RANK attribute is reinstated

The use of the DVAL_STATIC_RANK attribute has been reinstated with the MDEX Engine 6.1.4. This attribute belongs to the STATS element. It specifies whether every dimension value's static rank should be returned as a property on the dimension value. The default value is FALSE.

Setting this attribute to TRUE causes the MDEX Engine to return the static rank with each dimension value. Like other attributes in the STATS element configuration, the value for this attribute can be specified both at the individual dimension level and at the global level.

Note: This attribute has been deprecated in 6.1.0 - 6.1.3 releases of MDEX Engine. If this attribute was used in those releases, the MDEX Engine ignored it and issued a warning about its presence in the file.

Spelling correction can be disabled per query

You can to disable spelling correction and DYM suggestions on individual queries. This reduces the cost of running some queries in performance-sensitive applications.

In the presentation API, use the `spell+nospell` option with `Ntx` and `Dx` parameters. For example:

```
D=blue+suede+shoes&Dx=mode+matchallpartial+spell+nospell
```

In the Dgraph URL, specify the `spell+nospell` value to the `opts` parameter. For example, change this type of query from this syntax:

```
/search?terms=blue+suede+shoes&opts=mode+matchallpartial
```

To the following syntax:

```
/search?terms=blue+suede+shoes&opts=mode+matchallpartial+spell+nospell
```

In the Java Presentation API, you can disable spelling for a specific query as shown in this example:

```
ENEQuery nequery = new ENEQuery();
nequery.setDimSearchTerms("blue suede shoes");
nequery.setDimSearchOpts("spell nospell");
```

In the .NET API, you can disable spelling for a specific query as shown in this example:

```
ENEQuery nequery = new ENEQuery();
nequery.DimSearchTerms = "blue suede shoes";
nequery.DimSearchOpts = "spell nospell";
```

For more information on this option, see the chapter on spelling correction in the *MDEX Engine Advanced Developer's Guide*.

Dynamic refinement ranking of collapsible dimensions

This topic discusses the interaction of dynamic refinement ranking with collapsible dimensions. The `--dynrank_consider_collapsed` Dgraph flag forces the MDEX Engine to consider intermediate collapsible dimension values as candidates for dynamic ranking.

By default, the MDEX Engine considers only leaf dimension values for dynamic ranking, removing all intermediate dimension hierarchy from consideration. With this default behavior, when a hierarchical dimension's mid-level values (all except the root and leaf values) are configured as collapsible in Developer Studio, and when the dimension is also set to use dynamic refinement ranking, the dimension

collapses and displays only leaf values for all navigation queries. The mid-level dimension values are never displayed regardless of the number of leaf values present in the navigation state.

You can use the `--dynrank_consider_collapsed` flag to force the MDEX Engine to consider intermediate collapsible dimension values as candidates for dynamic ranking.

Record and dimension value boost

Two new features allow you to manipulate the rankings of returned records or dimension values.

Record boost and bury is a mechanism by which the ranking of certain specific records is made much higher or lower than other records. This allows you to manipulate ranking of results in order to push certain types of records to the top or bottom of the results list. The feature depends on the use of the new `stratify` relevance ranking module.

Dimension value boost and bury is a companion feature that allows you to re-order returned dimension values. With this feature, you can place dimension values into ranked strata, in order to promote or push down refinements. The feature depends on the use of the new `Nrcs` URL parameter and the related Presentation API methods.

See the *MDEX Engine Basic Development Guide* for details on these features.

Multiselect-OR improvements

The MDEX Engine's handling of multiselect-OR refinements has been improved.

Previously, dimensions were being labelled incorrectly as fully-implicit dimensions and therefore were not being put in the `DimensionList` returned by the Java `Navigation.getRefinementDimensions()` and .NET `Navigation.RefinementDimensions` calls. In 6.1.4, these calls return fully-implicit multiselect-OR dimensions where at least one selection has been made. This change also applies to the `Navigation.getRefinementDimGroups()` and `Navigation.RefinementDimGroups` calls.

In the context of this change, the definition of *implicit refinements* in the documentation has been updated to: Implicit refinements are refinements which, if selected, will not alter the navigation state record set.

Multiselect-OR sample scenarios

To illustrate the multiselect-OR changes, assume a multiselect-OR dimension named PriceRange that has three dimension values and has the following sample data:

```
Dimension values:
10+
20+
30+

Data:
productA, $5, ()
productB, $15, (10+)
productC, $25, (10+, 20+)
productD, $35, (10+, 20+, 30+)
```

This table shows the pre-6.1.4 and 6.1.4 results for various sample queries:

Query scenario	Pre-6.1.4 results	6.1.4 results
Root Node query. Submit an N=0 query with PriceRange exposed.	all PriceRange dimension values returned	all PriceRange dimension values returned
Select 10+ query. Submit a query with 10+ selected and PriceRange exposed.	• descriptors: 10+ • refinements (non-implicit): 20+, 30+ • refinements (implicit): none	• descriptors: 10+ • refinements (non-implicit): none • refinements (implicit): 20+, 30+
Select 20+ query. Submit a query with 20+ selected and PriceRange exposed.	• descriptors: 20+ • refinements (non-implicit): 30+ • refinements (implicit): 10+	• descriptors: 20+ • refinements (non-implicit): 10+ • refinements (implicit): 30+
Select 30+ query. Submit a query with 30+ selected and PriceRange exposed.	• descriptors: 30+ • dimension removed from normal section	• descriptors: 30+ • refinements (non-implicit): 10+, 20+ • refinements (implicit): none

MDEX Engine startup behavior for updates

The startup behavior for the MDEX Engine has changed when there are updates to be applied.

Previously, if there were partial update files in the updates directory when the MDEX Engine was started, it would first process all update files before opening the server port and accepting queries.

In 6.1.4, the MDEX Engine starts processing queries immediately, even when updates are found in the updates directory at startup time. In other words, the MDEX Engine's startup behavior is to process updates in parallel with queries.

Improvements in the default stemming dictionaries

MDEX Engine 6.1.4 includes new default stemming dictionaries for the following languages:

- German
- Spanish
- French
- Italian
- Portugese

The new dictionaries provide better search results because the dictionaries contain improved variants that are based on linguistic declination of proper nouns, nouns, and adjectives. (Prior releases included stemming dictionaries that had more simple rule-based variants.)

There were also minor improvements to the English stemming dictionary that are described in the release notes.

For details about stemming, see "Using Stemming and Thesaurus" in the *Endeca MDEX Engine Advanced Development Guide*.

Default stemming dictionaries can be supplemented

You can supplement the default stemming dictionaries by specifying a flag to Dgidx (`--stemming-updates`) and providing an XML file of custom stemming changes. The stemming update file specifies additions and deletions.

For usage details, see "Supplementing the default stemming dictionaries" in the *Endeca MDEX Engine Advanced Development Guide*.

Behavioral changes in version 6.1.3

This section lists all behavioral changes in version 6.1.3.

Support for VMware ESX 3.5

The MDEX Engine is supported on VMware ESX 3.5 for the following guest operating system platforms:

Platform	Recommended MDEX Engine server configuration
RHEL 5	• Configure four VCPUs on a virtual machine • Allocate a single Dgraph per virtual machine • Specify four threads for each Dgraph
Windows 2003	• Configure four VCPUs on a virtual machine • Allocate a single Dgraph per virtual machine • Specify four threads for each Dgraph

The number of threads should not exceed the number of VCPUs. Endeca does not recommend running more than one MDEX Engine per virtual machine.

Note: VMware configurations of the MDEX Engine are not supported with the Agraph.

For additional information on VMware support and performance, see the *Performance Tuning Guide*.

Support for Windows Server 2008

The MDEX Engine has added support for Windows Server 2008.

For best performance on Windows Server 2008, Endeca recommends the Enterprise Edition R2. If you experience poor performance running Windows Server 2008, see the *MDEX Engine Performance Tuning Guide* for more information.

Updates to the spelling dictionary allowed while running partial updates

A new administrative query operation has been added to the MDEX Engine, `admin?op=updatea¬spell`. You can use this operation to issue administrative queries that update the spelling dictionary while running partial updates. In previous releases, changes to the spelling dictionary required stopping and restarting the MDEX Engine.

If the amount of searchable text is large, this increases the latency of the `admin?op=updateaspell` operation.

Related Links

admin?op=updateaspell on page 43
In this release, you can update the `aspell` spelling dictionary in real time without restarting the MDEX Engine.

Deprecation of the --aspell flag of the dgwordlist utility on page 43
The `--aspell` flag of the `dgwordlist` utility has been deprecated and is ignored if specified.

admin?op=updateaspell

In this release, you can update the `aspell` spelling dictionary in real time without restarting the MDEX Engine.

Use the `admin?op=updateaspell` administrative operation to rebuild the aspell dictionary for spelling correction from the data corpus.

The `admin?op=updateaspell` operation performs the following actions:

- Crawls the text search index for all terms
- Compiles a text version of the `aspell` word list
- Converts this word list to the binary format required by `aspell`
- Causes the Dgraph to finish processing all existing preceding queries and temporarily stop processing incoming queries
- Replaces the previous binary format word list with the updated binary format word list
- Reloads the `aspell` spelling dictionary
- Causes the Dgraph to resume processing queries waiting in the queue

The Dgraph applies the updated settings without needing to restart.

Only one `admin?op=updateaspell` operation can be processed at a time.

The `admin?op=updateaspell` operation returns output similar to the following in the Dgraph error log:

```
...
aspell update ran successfully.
...
```

Note: If you start the Dgraph with the `-v` flag, the output also contains a line similar to the following:

```
Time taken for updateaspell, including wait time on any
previous updateaspell, was 290.378174 ms.
```

Agraph support

The `admin?op=updateaspell` is not supported in the Agraph.

Deprecation of the --aspell flag of the dgwordlist utility

The `--aspell` flag of the `dgwordlist` utility has been deprecated and is ignored if specified.

In previous releases, the `--aspell` flag of the `dgwordlist` utility specified the location of the `Aspell` dictionary indexing program.

Starting with this release, the `dgwordlist` utility has been redesigned and no longer needs to know the location of `Aspell`. The `--aspell` flag is ignored if specified.

Partial updates with duplicate properties cause warnings

Certain types of partial updates that succeeded with a warning in 6.1.1 failed with an error in 6.1.2.

In the release 6.1.2, these partial updates actually succeeded, but the Dgraph issued an error, and the update file was moved to the `failed_updates` directory.

The conditions when the Dgraph behaved this way were as follows:

- Adding a duplicate dimension value or property value to a record
- Deleting a dimension value assignment that does not exist on a record.

Starting with version 6.1.3, the Dgraph exhibits the same behavior as in version 6.1.1, and issues warnings for these conditions. No files are added in these cases to the `failed_updates` directory.

Refinement ranking of aggregated records

The MDEX Engine uses the aggregated record counts beneath a given refinement for its refinement ranking strategy only if they were computed for the query sent to the MDEX Engine.

The MDEX Engine computes refinement ranking based on statistics for the number of records beneath a given refinement. In the case of aggregated records, refinement ranking depends on whether you have requested the MDEX Engine to compute statistics for aggregated record counts beneath a given refinement.

The following statements describe the behavior:

- To enable dynamic statistics for aggregated records (aggregated record counts beneath a given refinement), use the `--stat-abins` flag with the Dgraph.
- To retrieve the aggregated record counts beneath a given refinement, use the `DGraph.AggrBins` property.
- If you specify `--stat-abins` when starting a Dgraph and issue an aggregated query to the MDEX Engine, it then computes counts for aggregated records beneath a given refinement, and generates refinement ranking based on statistics computed for aggregated records.
- If you specify `--stat-abins` and issue a non-aggregated query to the MDEX Engine, it only computes counts for regular records (instead of aggregated record counts) beneath a given refinement, and generates refinement ranking based on statistics computed for regular records.
- If you do not specify `--stat-abins` and issue an aggregated query to the MDEX Engine, it only computes counts for regular records (instead of aggregated record counts) beneath a given refinement, and generates refinement ranking based on statistics computed for regular records.

To summarize, the MDEX Engine uses the aggregated record counts beneath a given refinement for its refinement ranking strategy only if they were computed. In all other cases, it uses only regular record counts for refinement ranking.

Change to the MDEX API through XQuery

Users of the MDEX API through XQuery, or MAX, can now specify a per-query language ID for navigation, dimension search, and compound dimension search queries.

The data types `NavigationQuery`, `DimensionSearchQuery`, `DimensionSearchAppliedFil¬ters`, `CompoundDimensionSearchQuery`, and `CompoundDimensionSearchAppliedFilters` now include a `LanguageId` element.

Detailed information about MAX can be found in the *Web Services and XQuery Developer's Guide*.

Note: This change will not break existing client bindings to the MDEX Web service. You do not need to regenerate stubs unless you want to use the new feature.

Host and port are no longer accepted in the mdex?wsdl service

In version 6.1.2, in order to get an address returned in the port binding when calling the `mdex?wsdl` service, you needed to specify `host` and `port` arguments.

In version 6.1.3, you no longer need to specify these arguments. The `mdex?wsdl` service now pulls the `Host` header from the HTTP request directly and uses the `host` and `port` information contained in that header.

Change to the interaction between try/catch expressions and updating expressions

In previous versions, if updates were done within an XQuery try block and then an exception was thrown, the updates would still be applied.

Now, when try/catch expressions are used with the Endeca implementation of XQuery update, if exceptions are raised in a try block, any updates appended within that try block are removed from the pending update list. This rollback is applied regardless of whether the exception is caught at that point, caught further up the stream, or escapes the program.

Note: XQuery update, an Early Access feature in this release, is documented in the *Web Services and XQuery Developer's Guide*.

Example

For example, in this version, the following code sample puts y into the collection:

```
try {
     put(<x/>, "mdex://x")
     error(...)
}
catch {
     put(<y/>, ...)
}
```

However, in the previous version, it would have put both x and y into the collection.

Behavioral changes in version 6.1.2

This section lists all behavioral changes in version 6.1.2.

Running updates on a single file

In some cases, you may need to run a partial update by pointing the Dgraph to a single file by using the `admin?op=update&updatefile=filename` option.

The recommended way of running partial updates is by using the `admin?op=update` URL command that applies all files residing in the `dgraph_input/updates` directory (or the directory that you specify for updates with the `--updatedir` flag). However, pointing the Dgraph to a single updates file may be useful for performance testing purposes, such as when you plan to run Eneperf in the two-stream mode to test MDEX Engine performance with partial updates.

Note: Before running Eneperf in the two-stream mode, you first need to obtain a separate request log that contains only partial update requests issued to the MDEX Engine. You can obtain such a log when you run several partial updates on single update files. For more information on running Eneperf for testing updates performance, see the *Performance Tuning Guide*.

To run a partial update on a single file:

1. Add the update file to the `dgraph_input/updates` directory or the directory specified using the `--updatedir` flag.
2. In your Web browser, issue the update command with this URL syntax:

   ```
   http://hostname:dgraphport/admin?op=update&updatefile=filename
   ```

   where `filename` is the name of an update file residing in the updates directory.

   You can run this command on a single file only. If you have more than one file, rerun this command once for each file.

   The MDEX Engine deletes the update file after successfully applying the results of the partial update.

Note: For performance reasons, Endeca recommends running partial updates in batch mode, by only using the `admin?op=update` command. This command applies all the update files present in the `dgraph_input/updates` directory.

Using Eneperf in two-stream mode to test updates performance

Two new settings have been added to Eneperf, `--updates-log` and `--msec-between-updates`. Use these settings if you want to run Eneperf in a two-stream mode to test MDEX Engine performance with partial updates that are sent at regular intervals while Eneperf processes query requests.

To test updates performance with Eneperf, create a separate updates log, and use the `--updates-log` setting together with `log` and `--msec-between-updates` settings.

The new Eneperf settings are described as follows:

Setting	Description
`--updates-log`	Specify the updates log file that contains partial update requests to replay at every interval in milliseconds specified with `--msec-between-updates`. Specifying the updates log allows running Eneperf in a two-stream mode with two logs: regular query request logs and update request logs. In this mode, Eneperf sends update requests from the updates log at regular intervals while sending queries from the query log. This setting must be used together with `--msec-between-updates`. This setting must not be used together with `--list`, `--seek`, `--seekrepeat`, `--prelude`,`--postlude`, and `--throttle`. Before running Eneperf in the two-stream mode, you need to create a separate log that contains only partial update requests. You should create such a log with several partial update requests pointing to a single update file using the `admin?op=update&updatefile=filename` command. For more information on running partial updates on a single file, see the *Partial Updates Guide*.
`--msec-between-updates`	The minimum time interval between sending partial update requests, in milliseconds. Before sending a new update request, Eneperf waits for a free connection (after the specified time interval expires). This setting must be used together with `--updates-log`. This setting must not be used together with `--list`, `--seek`, `--seekrepeat`, `--prelude`,`--postlude`, and `--throttle`.

The format of the updates log is the same as the format of the regular query log for Eneperf, except that the updates log should contain only `config?op=update` operations in order to provide meaningful performance results. (If your updates log is similar to your regular log, Eneperf still runs on this log successfully, however the results are not useful to measure updates performance.)

Using `--updateslog` and `--log` settings is useful to measure performance of those updates that run at regular intervals. To test updates that run at random times, you can continue using your regular log with Eneperf.

Note: The actual time interval between sending update requests may be equal to or greater than the time specified with `--msec_between_updates`. This is because Eneperf uses the

same `num connections` setting while processing the regular query log and updates log. This causes Eneperf to wait for a preceding request to complete before it can process the next updates log request.

For detailed information about running Eneperf in the two-stream mode, see the Endeca *Performance Tuning Guide*.

Support for the cluster discovery feature restored

The cluster discovery feature is supported in the version 6.1.2 of the MDEX Engine (if you use the Presentation API).

Note: Term discovery and cluster discovery are part of the Endeca Relationship Discovery module. For more information on cluster discovery, see the *Relationship Discovery Guide*, which is part of the Endeca Platform Services installation package.

The MDEX Engine always runs in a multithreaded mode

In versions 6.1.x, to ensure better resource management, the MDEX Engine always runs in a multithreaded mode with the default number of threads equal to 1. In addition, starting with version 6.1.2, the `--threads 0` value is ignored and interpreted as `--threads 1`.

Multithreaded mode is the only supported mode for the MDEX Engine and it cannot be disabled.

If you set the `--threads 0` value, the Dgraph issues a warning and continues to run with one thread used for processing client requests (queries and partial updates).

As in previous releases, Endeca recommends that you experimentally increase the number of processing threads to improve performance. In addition to threads controlled by the threading pool (with the `--threads` flag), the MDEX Engine by default uses a pool of background threads (that you cannot control) .

The MDEX Engine threading pool

Starting with 6.1.2, the MDEX Engine consistently manages all processor-intensive tasks related to query and updates processing by using its preconfigured threading pool. The `--threads` flag reflects the total number of threads in the MDEX Engine threading pool.

With this change, the MDEX Engine consistently manages all CPU-intensive operations using its preconfigured threading pool. The threading pool controls the total number of all high-priority, query-related threads. These threads include query processing and partial update processing threads and additional threads that support query and update processing. Prior to MDEX Engine 6.1.2, the number of threads controlled by the `--threads` flag reflected only threads used for query processing.

You define the number of threads in the threading pool at MDEX Engine startup, based on the setting for the `--threads` flag.

Recall that the recommended number of threads for the MDEX Engine is typically equal to the number of cores on the MDEX Engine server. By managing the threading pool, the MDEX Engine lets you more accurately limit the available computation resources to each core. This ensures that the system resources are used effectively for the highly prioritized tasks in the MDEX Engine, all of which support query processing and high performance.

The threading pool manages the following MDEX Engine tasks:

- Query processing tasks.
- Update and administrative operations.
- All tasks that support query processing in the MDEX Engine. The MDEX Engine allocates these tasks for threads in the threading pool. The tasks include all high-priority, CPU-intensive, frequently performed operations the MDEX Engine runs in production. For example, they include precomputed sorting, background merging of index generations, and operations that support high performance of updates, among others.

Other MDEX Engine operations that do not have a significant impact on CPU usage are not managed by the threading pool.

Note: If you use operating system commands such as `top` to examine the number of threads used by the MDEX Engine server, you may see a number that is larger than the number you specify with the `--threads` flag. This is because in addition to this number of threads, the MDEX Engine may use additional threads for other tasks. These additional threads support tasks that are run infrequently, are less-CPU intensive, and do not affect overall MDEX Engine performance. You cannot control these additional threads.

Wildcard search simplification

The MDEX Engine 6.1.2 uses a new mechanism for processing wildcard search queries that greatly simplifies user configuration. In most cases, the size of the on-disk index is reduced considerably, and at the same time indexing performance is improved compared with previous releases.

The new mechanism replaces the regular- and dictionary-based wildcard search methods utilized in previous releases.

The following changes describe the new method and the differences with the previous releases:

- **Configuration**. The configuration to enable wildcard search remains the same as in previous releases. You should enable wildcard search using the Developer Studio. For configuration information, see the *MDEX Engine Basic Development Guide*, or the *Developer Studio Help*.
- **Fewer tuning settings**. The new wildcard search mechanism deprecates a number of command-line flags and attributes that existed previously for tuning purposes.

  The following wildcard configuration attributes in the `searchindex.dtd` are deprecated:

  - `MAX_NGRAM_LENGTH`
  - `DICTIONARY_WILDCARD`
  - `DICTIONARY_MAX_NGRAM_LENGTH`

  

  Note: Do not remove these settings from the XML configuration files since they continue to be part of the `searchindex.dtd` used to validate the interface between Developer Studio and Dgidx. If these attributes are present in the XML configuration files, the Dgraph ignores them during startup without issuing a warning.

- The following Dgraph and Dgidx flags are deprecated:
  - `--wildcard_approx` (Dgraph)
  - `--ngram_min` (Dgidx)
- **Performance**. For optimal performance, Endeca recommends using wildcard search queries with at least two or three non-wildcarded characters in them, such as `abc*` and `ab*de`, and avoiding

wildcard searches with one non-wildcarded character, such as `a*`. Wildcard queries with extremely low information, such as `a*`, require more time to process.

- **`wildcard_max` remains the only tuning option for wildcard search**. For the majority of wildcard search patterns, the MDEX Engine does not rely on `--wildcard_max` and you should not adjust it.

  The maximum number of matching terms of a wildcard expression is set to 100 by default. You can continue to modify this value with the `--wildcard_max` flag for the Dgraph to balance performance of the wildcard search with the desired completeness of results.

  Consider increasing this value only if you have wildcard search queries that use punctuation syntax, such as `ab*c.def*`, and you would like to receive more complete wildcard query results, and can afford slower running wildcard search queries in such cases. This value does not affect other wildcard search queries.

  If in previous releases you used `--wildcard_max` in cases other than the one described above, such as for `a*b*` queries (that do not contain punctuation), after upgrading to this release, consider resetting the value of this flag back to its default (100), and testing performance of the MDEX Engine (it should be improved). For detailed information about tuning `--wildcard_max`, see the *Performance Tuning Guide*.

Related Links

Deprecation of wildcard search settings and flags on page 50

The MDEX Engine ignores `MAX_NGRAM_LENGTH`, `DICTIONARY_MAX_NGRAM_LENGTH`, and `DICTIONARY_WILDCARD` settings in the XML configuration files. The `--wildcard_approx` Dgraph flag is deprecated and is ignored by the MDEX Engine. The `--ngram_min` Dgidx flag is deprecated and ignored.

Deprecation of wildcard search settings and flags

The MDEX Engine ignores `MAX_NGRAM_LENGTH`, `DICTIONARY_MAX_NGRAM_LENGTH`, and `DICTIONARY_WILDCARD` settings in the XML configuration files. The `--wildcard_approx` Dgraph flag is deprecated and is ignored by the MDEX Engine. The `--ngram_min` Dgidx flag is deprecated and ignored.

The following settings and flags for wildcard search have been deprecated or their usage has been changed:

Setting or flag that is deprecated in 6.1.2	Description
`MAX_NGRAM_LENGTH`	This setting is deprecated and ignored by the MDEX Engine, as it is no longer necessary for the wildcard search implementation. In previous releases, this setting represented the maximum substring length that was being indexed. It belongs to the `RECSEARCH_INDEXES` and `DIMSEARCH_INDEX` elements in the XML configuration files.
`DICTIONARY_WILDCARD`	This attribute is deprecated and ignored by the MDEX Engine. In previous releases, this attribute enabled dictionary-based wildcard search, and indicated whether the dictionary-based

Setting or flag that is deprecated in 6.1.2	Description
	index had to be created. The dictionary-based index is no longer used by the new wildcard mechanism. It belongs to the RECSEARCH_INDEXES and DIMSEARCH_IN¬DEX elements in the XML configuration files.
DICTIONARY_MAX_NGRAM_LENGTH	This setting is deprecated and ignored. In previous releases, this setting represented the maximum substring length that was indexed for the dictionary-based wildcard index. It belongs to the RECSEARCH_INDEXES and DIMSEARCH_IN¬DEX elements in the XML configuration files.
--wildcard_approx	This Dgraph flag is deprecated and ignored. The Dgraph issues a warning if it is specified. In previous releases, you could use this flag in some cases to improve performance of wildcard search by allowing approximate wildcard search query matching and not validating substring match results. The new wildcard method significantly reduces the complexity associated with post-filtering of the result set. This eliminates the need for this flag.
--ngram_min	This Dgidx flag is deprecated and ignored since it no longer applies to wildcard indexing. Dgidx issues a warning if it is specified.

The Dgraph checks permissions on the index directories

Starting with version 6.1.2, the Dgraph checks permissions on index directories before applying partial updates.

If the required read/write permissions are missing, the Dgraph fails to apply the update and issues an error in the standard error log. It also logs the path to the index directories to which it does not have read/write permissions.

The Dgraph checks permissions on these directories in the Endeca/myApp/dgidx_out¬put/myApp_indexes:

- /committed
- /generations

(The filepaths assume that the Deployment Template scripts are used to set up the application.)

Both of these directories should have read and write permissions to allow accessing them by the Dgraph. However, due to file system issues or hardware maintenance issues combined with the Endeca implementation's topology, it is possible that under some conditions these permissions are reset. This may make these directories unaccessible by the Dgraph.

Changes to supplemental objects returned by the MDEX Engine

After you upgrade to the MDEX Engine 6 series, you may notice that supplemental objects issued by the MDEX Engine in response to queries no longer return record properties in some cases. This is the expected and correct behavior

After you upgrade to the MDEX Engine version 6, the MDEX Engine still returns properties with lists of records, but the behavior is somewhat different:

- It returns only those properties for which you have specified **Show with Record List** in Developer Studio. In previous releases, it returned all properties for a record that were specified as **Show with Record**.

  As a result, fewer properties are now returned than before.

- It returns these properties in record lists returned in response to regular user queries, and also in record lists returned by the dynamic business rules. (Dynamic business rules enable merchandizing and content spotlighting.)

  In terms of XML configuration settings, rule results from the MDEX Engine now use the `REN¬DER_PROD_LIST` setting from the `RENDER_CONFIG.XML` file, rather than the `RENDER_PROD_PAGE` setting as they did in IAP 5.x and earlier versions.

Note: The previous behavior was a bug that has been adversely affecting performance.

This behavioral change may be important to you if you were using record properties returned by the MDEX Engine for the display of content spotlighting and merchandizing (dynamic business rules), or for other purposes.

After upgrading, if you were relying on all record properties returned by the MDEX Engine, you may want to test your Endeca application to verify that you are not missing properties required for rendering spotlighted records. (If some properties are missing, make sure they are configured to **Show with Record List**.)

Background information about record properties

The MDEX Engine typically returns additional information with a user query request. This is known as supplemental objects information. This information depends on the nature of the query.

For record properties, you can specify two options in the Property Editor of Developer Studio, **Show with Record** and **Show with Record List**. When you specify **Show with Record List**, the corresponding `RENDER_CONFIG.XML` file is updated. This file indicates to the MDEX Engine which mapped properties it must return as supplemental objects with the list of records.

Changes to the MDEX Engine Statistics page

Several items on the MDEX Engine Statistics page have changed, in response to support for different MDEX Engine features. The sections in this topic describe the changes in detail.

Replacement for the Performance Statistics section

The Performance Statistics section on the **Performance Summary** tab has been removed. Most of this section's metrics have been moved to the Server and Results sections of the **Details** tab.

Note: In version 6.1.2, the Server section of the **Details** tab is what used to have been titled the XQuery Server section in previous releases.

The following table lists those metrics from the Performance Statistics section of the **Performance Summary** tab that moved to the **Details** tab on the MDEX Engine Statistics page:

Metrics and their location in versions before 6.1.2	**Metrics and their location in version 6.1.2**
Location in versions before 6.1.2: **Performance Summary** > **Performance Statistics**	Location in version 6.1.2: **Details** > **Server**
Metrics name: Queue length	Metrics name: Scheduler: Queries queued
Metrics name: Number of threads busy	Metrics name: Scheduler: Queries in process
Metrics name: Total processing time	Metrics name: HTTP: Total request time
Metrics name: Response size (in bytes)	Metrics name: HTTP: Response size
Location in versions before 6.1.2: **Performance Summary** > **Performance Statistics** Metrics name: Number of records in result set	Location in version 6.1.2: **Details** > **Results** Metrics name: Number of records in result set

Changes to refinement generation tracking for clustering

Previously, refinement generation in support of the clustering feature was accounted for in the "Navigation - drill-downs" and "Navigation - drill-down refinement record counts" statistics elements, which are located in the Hotspots section of the **Details** tab. In this release, to distinguish between the costs associated with normal refinements and those in support of clustering, a new "Clustering performance" element has been added to the Hotspots section of the Details tab.

The "Clustering performance" element contains details about clustering refinements, clustering refinement record counts, and the time spent making clusters.

The "Navigation - drill-downs" and "Navigation - drill-down refinement record counts" elements still exist but no longer include costs associated with clustering.

Addition of XQuery Update Totals section on the Index Preparation tab

The I**ndex Preparation** tab now contains a separate section to track statistics for XQuery updates.

Note: The XQuery update feature is Early Access for version 6.1.2. For details, see the *Web Services and XQuery Developer's Guide*.

Changes to the Agraph Statistics page

The statistics metrics for `Num Requeries` and `Num Requeries Failed` are added to the Statistics page for the Agraph.

Note: These metrics are intended for internal use by Endeca Support. They have been added to support continuous query operations with the Agraph. For more information, see the "Agraph and continuous query support" topic in this guide.

For example, the following generalized excerpt from the Agraph Statistics page contains these new entries (highlighted):

```
Current time: [date and time]
Avg. Throughput (10 sec.):[req/sec]
Avg. Throughput (1 min.): [req/sec]
```

```
Avg. Throughput (5 min.): [req/sec]

General Information
Version: version [##]
PID: [#]
UID: [#]
GID: [#]
CWD: [/path_to_current_working_directory]
Host: [host_name]
Server Port: [8888]
Start Time: [date and time]
Data Path: [/path_to_the_agidx_data]
Data Tag: unknown
Data Date: [date and time]
Last Request Time: [date and time]
Num Requests: 6
Num Requeries: 2
Num Requeries Failed: 0
current_process_size (MB): 5.6523
```

Detailed documentation about the MDEX Engine Statistics page (for the Dgraph and the Agraph) can be found in the *Performance Tuning Guide*.

The Dgraph -A flag is deprecated

In previous releases, you could use the `dgraph -A` flag to disallow server shutdown and restart through `admin?op=exit` and `admin?op=restart` URL commands sent to the Dgraph.

The `dgraph -A` flag is deprecated starting with version 6.1.2 and is not guaranteed to be supported in future releases.

The Agraph and continuous query support

Starting with the MDEX Engine 6.1.2, the Agraph supports the MDEX Engine feature known as continuous query.

The following statements describe how the Agraph supports continuous query:

- The Agraph can continually answer queries to its clients even while partial updates are applied across its different child Dgraphs. This eliminates the need to stop the Agraph when applying partial updates to the Dgraphs.
- Although an Agraph does not wait for all of its child Dgraphs to finish updating when querying them, it always ensures that it uses consistent results from a single child Dgraph to which partial updates are being applied.

  A query result is always returned by the Agraph after it aggregates the child Dgraph results, without specific guarantees that all child results reflect the partial updates.
- If a deployment requires consistent results from all child Dgraphs following a partial update, Endeca recommends that you run multiple Agraphs (each with its set of child Dgraphs) within a load-balancer pool. This allows the selective removal of an Agraph from the pool, and lets pending requests complete before applying the updates to all its child Dgraphs. Once updates have been applied to the child Dgraphs, you can add the Agraph back to the load-balancer pool.

  In Agraph implementations that utilize a load balancer between a single Agraph and its Dgraphs, if a deployment requires consistent results from all child Dgraphs following a partial update, ensure

that re-queries made by an Agraph target the same child Dgraph. You can achieve this by configuring the load balancer to track session information, and ensuring that all requests associated with a session go to the same child Dgraph.

Changes to the MDEX API through XQuery

This topic provides a summary of the changes to the MDEX API through XQuery (or MAX).

- Analytics is now supported through the MAX API.
- User profiles are now supported through the MAX API.
- In the `BusinessRuleList` data type, in the `BusinessRule` element, `maxOccurs` has been changed from `unbounded` to `1`.
- The `Property` data type can now be an empty string.

Detailed information about these changes can be found in the *Web Services and XQuery Developer's Guide*.

Note: These changes will not break existing client bindings to the MDEX Web service. You do not need to regenerate stubs unless you want to use the new features.

Behavioral changes in XQuery

This section discusses behavioral changes in XQuery.

For details on any of these changes, see the *Web Services and XQuery Developer's Guide*.

Changes to the behavior of fn:error()

The behavior of the XQuery function `fn:error()` has changed in version 6.1.2.

In earlier versions, if an exception was thrown for any reason, an HTML response with the details of the exception was returned, with a status code of `500: Internal Server Error`.

In the current version, the third argument in the three-argument version of `fn:error()` is used to specify the error sequence. If an error raised by `fn:error()` is not caught, the contents of the third argument, if any, are serialized and returned as the body of the HTTP response. In the case of SOAP faults generated by the MDEX Web service, this appears in the `ErrorSequence` element of the `mdata:Fault`.

Change in error handling in Web services and XQuery for Endeca

The way errors are caught, handled, and reported in Web services and XQuery for Endeca has changed in version 6.1.2.

in previous releases, all exceptions generated the same message, EXTF0001. In this there are a number of more detailed error codes. These Endeca-specific codes supplement those provided by and explained in the XQuery specification. In addition, you can now use try/catch expressions for custom error handling.

Changes to the Query Web service

This section discusses changes to the Query Web service.

For details on any of these changes, see the *Web Services and XQuery Developer's Guide*.

Name change from Query Web service to MDEX Web service

In previous releases, the MDEX Web service was know as the Query Web service.

Error handling in the MDEX Web service

In earlier versions, the MDEX Web service had incomplete error handling. Web services that failed because of exceptions thrown by external functions returned a SOAP fault that included only the description field associated with the exception, and all SOAP Faults were classified as server errors.

Now, when the MDEX Web service encounters a runtime error, it can catch, package, and return the error.

These changes may break existing Web service clients that depend on the SOAP Fault Schema.

Behavioral changes in versions 6.1.0 or 6.1.1

This section lists behavioral changes in versions 6.1.0 and 6.1.1.

Improved XQuery performance

The performance of XQuery for Endeca has improved significantly in this release.

Expanded MDEX Engine HTTP support

With release 6.1.0, the MDEX Engine supports HTTP 1.0 and HTTP 1.1 clients.

However, it does not implement optional HTTP 1.1 features, such as Keep-Alive.

Ability to use Presentation API and Web services features at the same time

In version 6.0.1, the MDEX Engine ran in either Presentation API mode or Web services mode, depending on how it was started.

In this release, the features of both modes are included in MDEX Engine by default, and it is possible to use features of both at the same time. This means that you can add XQuery for Endeca functionality to an existing Presentation API based Endeca application. The flag that was formerly used to specify Web services mode, `--ws`, has been deprecated and will generate a warning message if used.

Impact on startup time

Because the MDEX Engine now loads the XQuery modules at startup, MDEX Engine startup takes longer than it did in version 6.0.1. In most cases, this is not an issue. However, if you find it a problem and are not planning to use Web services, you can avoid this startup time cost by starting the MDEX Engine with the `--disable_web_services` flag. This runs the MDEX Engine without Web services functionality.

Changes to the MDEX API through XQuery interface

This topic discribes changes to the MDEX API through XQuery (or MAX) interface.

Top level result elements used to be named `Results`. They are now named `<QueryType>Results` (for example, `NavigationResults`, `DimensionSearchResults`, and so on).

The `SortList` element's `RelevanceRanking` sub-element used to be of type `Search`. It is now of the new type `RelevanceRanking`.

Changes to admin operation support

The operations `/admin?op=update` and `/admin?op=updatehistory` requests are now supported using a Web services enabled MDEX.

Continuous query

Starting with version 6.1.0, the MDEX Engine processes partial updates concurrently with processing query requests.

During continuous query processing, the MDEX Engine Dgraph port remains open for both query processing and partial updates processing. (In previous releases, when processing partial updates, the MDEX Engine closed its port temporarily.)

Continuous query is enabled starting with the MDEX Engine version 6.1.0 for all types of queries to the Engine, including navigation, record and aggregated record queries, queries with text search, queries that contain filters (EQL, range and record filters), queries containing Web services and XQuery, and all other types of queries.

Since the MDEX Engine continues to process all incoming queries while partial updates are running, queries are processed against either the pre-update or post-update state of the index data, depending on when they arrive. Pre-update and post-update states refer to the states before and after a partial update was applied. The MDEX Engine never processes queries against the data that is in the state of being updated through a partial update.

With continuous query, the MDEX Engine maintains its query processing performance levels, including low query latency and partial updates latency.

A few administrative queries are processed differently; for details see the section "Continuous query processing and administrative queries".

In previous releases, you could specify the `offline=[true|false]` option as part of the `admin?op=update` query. This parameter has been removed and does not exist in the MDEX Engine 6.1.0, because the Engine no longer goes offline while processing updates. If you issue an `admin?op=update&offline=true|false`, the MDEX Engine ignores this request, issues a warning and continues to process an update while keeping its port open.

In addition, in previous releases, to ensure adequate performance after an update, after a partial update was complete and before opening its port to accept new queries, the MDEX Engine ran warming replay queries by default. Starting with version 6.1.0, the MDEX Engine no longer replays warming queries after updates since its port never closes during updates processing, and the `warmupseconds` option of the `admin?op=update` is ignored by the MDEX Engine. The Dgraph issues a warning if it is issued and continues its processing.

Continuous query processing and administrative queries

You can issue administrative queries to the MDEX Engine concurrently with running updates, without any interruptions caused by partial updates processing, except for the following administrative and configuration queries that are processed differently.

- `admin?op=exit`
- `admin?op=restart`
- `admin?op=reload-services`
- `config?op=update`

`admin?op=exit` and `admin?op=restart` queries cause the MDEX Engine to close its Dgraph port for accepting future queries. Next, the MDEX Engine processes all previously received queries and shuts down (or restarts, depending on which of these two commands is issued).

`config?op=update` and `admin?op=reload-services` operations cause the MDEX Engine to drain all existing preceding queries, temporarily stop processing other queries and begin to process `config?op=update` and `admin?op=reload-services`. After it finishes processing these operations, the MDEX Engine resumes processing queries that queued up temporarily behind these requests.

Only one `config?op=update` operation can be processed at a time.

Note: `config?op=update` and `admin?op=reload-services` can be time-consuming operations. This depends on the number of configuration files the MDEX Engine has to process for an update (during `config?op=update`), or the number of XQuery modules that you have created and that have to be compiled (during `admin?op=reload-services`).

You can issue all other administrative queries to the MDEX Engine concurrently with updates, without any interruptions caused by partial updates processing.

Deprecation of the DVAL_STATIC_RANK attribute

The `DVAL_STATIC_RANK` attribute is deprecated starting with the MDEX Engine 6.1.0. Note that this attribute is reinstated in the MDEX Engine 6.1.4.

If you specify the value for this attribute in releases 6.1.0 - 6.1.3, the Dgraph ignores it and issues a warning if it is found. Starting with the MDEX Engine 6.1.4, the MDEX Engine uses this attribute again.

In versions of the Endeca IAP prior to 6.1.0, you could specify the `DVAL_STATIC_RANK` attribute on the `STATS` element in the XML configuration files.

The `STATS` element instructs the MDEX Engine to return statistics about refinements (dimension values) as part of the search query results.

The `DVAL_STATIC_RANK` attribute of the `STATS` element specified whether every dimension value's static rank had to be returned as a property on the dimension value. The default value was `FALSE`.

Deprecation of the ENABLE_AUTO_SUGGEST and ENABLE_DID_YOU_MEAN attributes

The `ENABLE_AUTO_SUGGEST` and `ENABLE_DID_YOU_MEAN` attributes of the `RECSEARCH` element are deprecated and ignored by the Dgraph in the MDEX Engine 6.1.0.

If you edit the `TRUE` and `FALSE` values of these attributes in the XML configuration files, the Dgraph ignores these attributes and issues an error that specifies that these attributes are no longer supported.

The attributes themselves continue to be part of the Endeca DTDs and must be present in the XML configuration files.

Note: To enable automatic suggestion and "Did you mean" functions, you can continue to use the `--spl` and `--dym` flags with the Dgraph as in previous releases.

Changes to the MDEX Engine request log Total Request Lifetime field

In 6.0.1, the request lifetime tracked in the Total Request Lifetime field of the MDEX Engine request log ended when the connection was closed.

If connection close did not time out, this lifetime would include the time to transport the response to the client, and the time for the client to read the response. In 6.1.0, the request lifetime ends when the response has been successfully delivered to the socket layer.

Installation-related changes

This section describes some changes to the Windows installation process.

Changes to the Windows installation path

In version 6.0.x of the MDEX Engine, the Windows installer appended `MDEX\<version>` to whatever path you installed to.

If you installed to the default location, `C:\Endeca`, the installer created the directory structure `C:\Endeca\MDEX\<version>`. If you installed to `C:\Workspace`, the installer would create the directory structure `C:\Workspace\MDEX\<version>`.

In version 6.1.x, the default Windows install location is `C:\Endeca\MDEX\<version>`. If you choose to install to another location, the installer does not append `MDEX\<version>` to the path you choose. It installs directly into the directory you specify.

Windows installer treatment of non-empty destination directories

In version 6.1, it is not possible to install the MDEX Engine into a non-empty destination directory.

If you attempt to install the MDEX Engine to a non-empty destination directory, the installation shows an information message and then returns to the screen where you can select a new destination directory.

If you are running a silent installation and attempt to install to a non-empty destination directory, the installation fails. If you run the silent installer from the command line with the logging option, `/l=<path>`, the reason for the failure is recorded in the log.

Documentation changes in version 6.1

This section outlines changes to the MDEX Engine documentation set and delivery in version 6.1.

Reduction of the installed documentation set for the MDEX Engine

As of 6.1, the documentation installed with the MDEX Engine component has been reduced to include only the Licensing Guide and the release notes. All other documentation is available on the Endeca Developer Network (EDeN) for viewing or download.

New MDEX Engine Migration Guide

A new *MDEX Engine Migration Guide* has been added to the MDEX Engine documentation set. Previously, all Endeca components shared a single *Migration Guide*.

New MDEX Engine Partial Updates Guide

A new *MDEX Engine Partial Updates Guide* has been added to the MDEX Engine documentation set.

The bulk of the content in this guide could previously be found in the *Forge Guide*.

Basic and Advanced Development Guides

Starting with version 6.1.0, the documentation for the MDEX Engine includes a *Basic Development Guide* and an *Advanced Development Guide*. The contents of these two guides are roughly equal to the single *Endeca Developer's Guide* in the Endeca IAP version 5.1.x.

Previously, there were separate versions of the *Endeca Developer's Guide* for Java and .NET. The language-specific content has been combined in the two new guides.