

Namespace Senzing.Sdk

Classes

[SzBadInputException](#)

Defines an exceptional condition when an invalid input value is provided to a Senzing operation preventing the successful completion of that operation.

[SzConfigRetryable](#)

Used to annotate which Senzing SDK methods are dependent upon the active configuration being current.

[SzConfigurationException](#)

Defines an exceptional condition when a failure has occurred pertaining to the Senzing configuration.

[SzDatabaseConnectionLostException](#)

Extends [SzRetryableException](#) to define an exceptional condition where a database connection was lost causing a Senzing operation to fail. Retrying the operation would likely result in the connection being reestablished and the operation succeeding.

[SzDatabaseException](#)

Extends [SzUnrecoverableException](#) to define an exceptional condition triggered by a database error from which we cannot recover (e.g.: missing or unexpected schema definition).

[SzDatabaseTransientException](#)

Extends [SzRetryableException](#) to define an exceptional condition where an operation failed because a database condition that is transient and would like be resolved on a repeated attempt. Retrying the operation may result in it completing successfully.

[SzEnvironmentDestroyedException](#)

Extends [InvalidOperationException](#) so the exceptional condition of the [SzEnvironment](#) already being destroyed can be differentiated from other [InvalidOperationException](#) instances that might be thrown.

[SzException](#)

Defines the base exception for Senzing errors. This adds a property for the numeric Senzing error code which can optionally be set.

[SzFlags](#)

Provides aggregate [SzFlag](#) constants as well as extension methods and utility methods pertaining to [SzFlag](#) and [SzFlagUsageGroup](#).

[SzLicenseException](#)

Extends [SzUnrecoverableException](#) to define an exceptional condition triggered by an invalid, expired or exhausted Senzing license.

[SzNotFoundException](#)

Extends [SzBadInputException](#) to define an exceptional condition where the provided bad input to a Senzing operation is an identifier that could not be used to successfully locate required data for that operation.

[SzNotInitializedException](#)

Extends [SzUnrecoverableException](#) to define an exceptional condition triggered by Senzing not being initialized.

[SzReplaceConflictException](#)

Describes an exceptional condition when an attempt is made to replace a Senzing value with a new value providing it has not already been changed, however, the current value is no longer the expected value and has therefore already been changed.

[SzRetryTimeoutExceededException](#)

Extends [SzRetryableException](#) to define an exceptional condition where an operation failed because a timeout was exceeded. Retrying the operation (possibly with a longer timeout) may result in it completing successfully.

[SzRetryableException](#)

Defines an exceptional condition where the failure is an intermittent condition and the operation may be retried with the same parameters with an expectation of success.

[SzUnhandledException](#)

Extends [SzUnrecoverableException](#) to define an exceptional condition caused by an otherwise unhandled and unexpected failure in the Senzing SDK.

[SzUnknownDataSourceException](#)

Extends [SzBadInputException](#) to define an exceptional condition where the provided bad input to a Senzing operation is an identifier that could not be used to successfully locate required data for that operation.

[SzUnrecoverableException](#)

Defines an exceptional condition where the failure is not recoverable and all operations should be stopped until the system can be modified to resolve the condition causing the failure.

Interfaces

[SzConfig](#)

Defines the C# interface that encapsulates and represents a Senzing configuration and provides functions to operate on that configuration.

[SzConfigManager](#)

Defines the C# interface to the Senzing config management functions.

[SzDiagnostic](#)

Defines the interface to the Senzing diagnostic functions.

[SzEngine](#)

Defines the interface to the Senzing engine functions.

[SzEnvironment](#)

Provides a factory interface for obtaining the references to the Senzing SDK singleton instances that have been initialized.

[SzProduct](#)

Defines the C# interface to the Senzing product functions.

Enums

[SzFlag](#)

Enumerates the Senzing flag values so they can be referred to as bitwise enumerated flags.

[SzFlagUsageGroup](#)

Enumerates the various classifications of usage groups for the [SzFlag](#) instances.

Class SzBadInputException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Defines an exceptional condition when an invalid input value is provided to a Senzing operation preventing the successful completion of that operation.

```
public class SzBadInputException : SzException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← SzBadInputException

Implements

[ISerializable](#)

Derived

[SzNotFoundException](#), [SzUnknownDataSourceException](#)

Inherited Members

[SzException.ErrorCode](#), [Exception.GetBaseException\(\)](#),
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#), [Exception.GetType\(\)](#),
[Exception.ToString\(\)](#), [Exception.Data](#), [Exception.HelpLink](#), [Exception.HResult](#),
[Exception.InnerException](#), [Exception.Message](#), [Exception.Source](#),
[Exception.StackTrace](#), [Exception.TargetSite](#), [Exception.SerializeObjectState](#),
[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#)

Constructors

SzBadInputException()

Default constructor.

```
public SzBadInputException()
```

SzBadInputException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzBadInputException(Exception cause)
```

Parameters

cause [Exception](#)

The underlying cause for the exception.

SzBadInputException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzBadInputException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzBadInputException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzBadInputException(long? errorCode, string message, Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzBadInputException(string)

Constructs with a message explaining the reason for the exception.

```
public SzBadInputException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzBadInputException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzBadInputException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Interface SzConfig

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Defines the C# interface that encapsulates and represents a Senzing configuration and provides functions to operate on that configuration.

```
public interface SzConfig
```

Examples

Create from template configuration:

```
// How to create an SzConfig instance representing the template configuration
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the SzConfigManager instance
    SzConfigManager configMgr = env.GetConfigManager();

    // create the config from the template
    SzConfig config = configMgr.CreateConfig();

    // do something with the SzConfig
    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to create new SzConfig from the template.", e);
}
```

Create from configuration definition:

```
// How to create an SzConfig instance representing a specified config definition
try
{
    // obtain the SzEnvironment (varies by application)
```

```

SzEnvironment env = GetEnvironment();

// get the SzConfigManager instance
SzConfigManager configMgr = env.GetConfigManager();

// obtain a JSON config definition (varies by application)
string configDefinition = ReadConfigFile();

// create the config using the config definition
SzConfig config = configMgr.CreateConfig(configDefinition);

// do something with the SzConfig
. . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to create a new SzConfig from a definition.", e);
}

```

Create from registered configuration ID:

```

// How to get a config definition by its configuration ID
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the config manager
    SzConfigManager configMgr = env.GetConfigManager();

    // get a valid configuration ID (will vary by application)
    long configID = configMgr.GetDefaultConfigID();

    // get the config definition for the config ID
    SzConfig config = configMgr.CreateConfig(configID);

    // do something with the SzConfig
    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception
}

```

```
    LogError("Failed to Create SzConfig from config ID.", e);  
}
```

Remarks

The Senzing config functions provide means to create, manipulate and export Senzing JSON configurations.

An [SzConfig](#) instance is typically obtained from an [SzConfigManager](#) instance via one of the following methods:

- [CreateConfig\(\)](#)
- [CreateConfig\(string\)](#)
- [CreateConfig\(long\)](#)

Methods

Export()

Retrieves the definition for this configuration.

```
string Export()
```

Returns

[string](#)

The configuration definition formatted as a JSON object.

Examples

Usage:

```
// How to export config JSON from a config handle  
try  
{  
    // obtain the SzEnvironment (varies by application)  
    SzEnvironment env = GetEnvironment();  
  
    // get the SzConfigManager instance  
    SzConfigManager configMgr = env.GetConfigManager();
```

```
// get an SzConfig object (varies by application)
SzConfig config = configMgr.CreateConfig();

// export the config
string configDefinition = config.Export();
demoResult = configDefinition; // @replace
. . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to export configuration.", e);
}
```

Example Result:

The example result is rather large, but can be viewed [here](#) (formatted for readability).

Remarks

NOTE: Typically, an implementation's `ToString()` function will be implemented to return the result from this function.

Exceptions

[SzException](#)

If a failure occurs.

See Also

[Code Snippet: Register Data Sources](#), [Code Snippet: Initialize Config](#)

GetDataSourceRegistry()

Gets the data source registry for this configuration.

```
string GetDataSourceRegistry()
```

Returns

[string](#)

The data source registry describing the data sources for this configuration formatted as a JSON object.

Examples

Usage:

```
// How to get the data source registry from an in-memory config
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the SzConfigManager instance
    SzConfigManager configMgr = env.GetConfigManager();

    // get an SzConfig object (varies by application)
    SzConfig config = configMgr.CreateConfig();

    // get the data sources
    string registry = config.GetDataSourceRegistry();
    demoResult = registry; // @replace
    // do something with the returned JSON (e.g.: parse it and extract values)
    JsonObject? jsonObj = JsonNode.Parse(registry)?.AsObject();

    JSONArray? jsonArr = jsonObj?["DATA_SOURCES"]?.AsArray();

    // iterate over the data sources
    if (jsonArr != null)
    {
        for (int index = 0; index < jsonArr.Count; index++)
        {

            JsonObject? sourceObj = jsonArr[index]?.AsObject();

            string? dataSourceCode = sourceObj?["DSRC_CODE"]?.GetValue<string>();

            . . .

        }
    }

}
catch (SzException e)
{
    // handle or rethrow the exception
}
```

```
        LogError("Failed to get data sources.", e);
    }
```

Example Result: (formatted for readability)

```
{
  "DATA_SOURCES": [
    {
      "DSRC_ID": 1,
      "DSRC_CODE": "TEST"
    },
    {
      "DSRC_ID": 2,
      "DSRC_CODE": "SEARCH"
    }
  ]
}
```

Exceptions

[SzException](#)

If a failure occurs.

RegisterDataSource(string)

Adds a data source to this configuration.

```
string RegisterDataSource(string dataSourceCode)
```

Parameters

`dataSourceCode` [string](#)

The data source code for the new data source.

Returns

[string](#)

The JSON `string` describing the data source was registered.

Examples

Usage:

```
// How to register data sources with an in-memory config
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the SzConfigManager instance
    SzConfigManager configMgr = env.GetConfigManager();

    // get an SzConfig object (varies by application)
    SzConfig config = configMgr.CreateConfig();

    // add data sources to the config
    config.RegisterDataSource("CUSTOMERS");
    config.RegisterDataSource("EMPLOYEES");
    config.RegisterDataSource("WATCHLIST");

    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to add data sources.", e);
}
```

Example Result: (formatted for readability)

```
{
    "DSRC_ID": 1003
}
```

Remarks

Because [SzConfig](#) is an in-memory representation, the repository is not changed unless the configuration is [exported](#) and then [registered](#) via [SzConfigManager](#).

An exception is thrown if the data source already exists in the configuration.

Exceptions

[SzException](#)

If a failure occurs.

See Also

[Code Snippet: Register Data Sources](#)

UnregisterDataSource(string)

Removes a data source from this configuration.

```
void UnregisterDataSource(string dataSourceCode)
```

Parameters

dataSourceCode [string](#)

The data source code that identifies the data source to delete from the configuration.

Examples

Usage:

```
// How to unregister a data source from an in-memory config
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the SzConfigManager instance
    SzConfigManager configMgr = env.GetConfigManager();

    // get an SzConfig object (varies by application)
    SzConfig config = configMgr.CreateConfig();

    // delete the data source from the config
    config.UnregisterDataSource("CUSTOMERS");

    . . .

}
```

catch (SzException e)

```
{
```

```
// handle or rethrow the exception  
LogError("Failed to delete data source.", e);  
}
```

Remarks

Because [SzConfig](#) is an in-memory representation, the repository is not changed unless the configuration is [exported](#) and then [registered](#) via [SzConfigManager](#).

NOTE: This method is idempotent in that it succeeds with no changes being made when specifying a data source code that is not found in the registry.

WARNING: If records in the repository refer to the unregistered data source, the configuration cannot be used as the active configuration.

Exceptions

[SzException](#)

If a failure occurs.

Interface SzConfigManager

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Defines the C# interface to the Senzing config management functions.

```
public interface SzConfigManager
```

Examples

An `SzConfigManager` instance is typically obtained from an [SzEnvironment](#) instance via the [GetConfigManager\(\)](#) method as follows:

```
// How to obtain an SzConfigManager instance
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the config manager
    SzConfigManager configMgr = env.GetConfigManager();

    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to get SzConfigManager.", e);
}
```

Methods

CreateConfig()

Creates a new [SzConfig](#) instance from the template configuration definition.

```
SzConfig CreateConfig()
```

Returns

[SzConfig](#)

A newly created [SzConfig](#) instance representing the template configuration definition.

Examples

Usage:

```
// How to create an SzConfig instance representing the template configuration
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the SzConfigManager instance
    SzConfigManager configMgr = env.GetConfigManager();

    // create the config from the template
    SzConfig config = configMgr.CreateConfig();

    // do something with the SzConfig
    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to create new SzConfig from the template.", e);
}
```

Exceptions

[SzException](#)

If a failure occurs.

See Also

[Code Snippet: Register Data Sources](#), [Code Snippet: Initialize Config](#)

[CreateConfig\(long\)](#)

Creates a new [SzConfig](#) instance for a configuration ID.

```
SzConfig CreateConfig(long configID)
```

Parameters

configID [long](#)

The configuration ID of the configuration to retrieve.

Returns

[SzConfig](#)

A newly created [SzConfig](#) instance representing the configuration definition that is registered with the specified config ID.

Examples

Usage:

```
// How to get a config definition by its configuration ID
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the config manager
    SzConfigManager configMgr = env.GetConfigManager();

    // get a valid configuration ID (will vary by application)
    long configID = configMgr.GetDefaultConfigID();

    // get the config definition for the config ID
    SzConfig config = configMgr.CreateConfig(configID);

    // do something with the SzConfig
    //

}

catch (SzException e)
{
    // handle or rethrow the exception
}
```

```
        LogError("Failed to Create SzConfig from config ID.", e);
    }
```

Remarks

If the configuration ID is not found the an exception is thrown.

Exceptions

[SzException](#)

If a failure occurs.

CreateConfig(string)

Creates a new [SzConfig](#) instance from a configuration definition.

```
SzConfig CreateConfig(string configDefinition)
```

Parameters

[configDefinition](#) [string](#) ↗

The definition for the Senzing configuration.

Returns

[SzConfig](#)

A newly created [SzConfig](#) representing the specified configuration definition.

Examples

Usage:

```
// How to create an SzConfig instance representing a specified config definition
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();
```

```

// get the SzConfigManager instance
SzConfigManager configMgr = env.GetConfigManager();

// obtain a JSON config definition (varies by application)
string configDefinition = ReadConfigFile();

// create the config using the config definition
SzConfig config = configMgr.CreateConfig(configDefinition);

// do something with the SzConfig
.

.

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to create a new SzConfig from a definition.", e);
}

```

Exceptions

[SzException](#)

If a failure occurs.

GetConfigRegistry()

Gets the configuration registry.

```
string GetConfigRegistry()
```

Returns

[string](#)

The JSON object [string](#) describing the configurations registered in the repository with their identifiers, timestamps and comments.

Examples

Usage:

```

// How to get a JSON document describing all registered configs
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the config manager
    SzConfigManager configMgr = env.GetConfigManager();

    // get the config definition for the config ID
    string registry = configMgr.GetConfigRegistry();
    demoResult = registry; // @replace
    // do something with the returned JSON (e.g.: parse it and extract values)
    JsonObject? jsonObj = JsonNode.Parse(registry)?.AsObject();

    JSONArray? jsonArr = jsonObj?["CONFIGS"]?.AsArray();

    // iterate over the registered configurations
    if (jsonArr != null)
    {
        for (int index = 0; index < jsonArr.Count; index++)
        {
            JsonObject? configObj = jsonArr[index]?.AsObject();

            long? configID = configObj?["CONFIG_ID"]?.GetValue<long>();
            string? createdOn = configObj?["SYS_CREATE_DT"]?.GetValue<string>();
            string? comment = configObj?["CONFIG_COMMENTS"]?.GetValue<string>();

            . . .

        }
    }
}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to get configurations.", e);
}

```

Example Result:

(formatted for readability)

```
{
  "CONFIGS": [
    {

```

```

    "CONFIG_ID": 48641147,
    "CONFIG_COMMENTS": "Data Sources: EMPLOYEES",
    "SYS_CREATE_DT": "2025-10-17T23:20:48Z"
},
{
    "CONFIG_ID": 181306537,
    "CONFIG_COMMENTS": "Initial Config",
    "SYS_CREATE_DT": "2025-10-17T23:20:48Z"
},
{
    "CONFIG_ID": 1433495806,
    "CONFIG_COMMENTS": "Data Sources: VIPS",
    "SYS_CREATE_DT": "2025-10-17T23:20:48Z"
},
{
    "CONFIG_ID": 1588422919,
    "CONFIG_COMMENTS": "Added PASSENGERS data source",
    "SYS_CREATE_DT": "2025-10-17T23:20:48Z"
},
{
    "CONFIG_ID": 3202498516,
    "CONFIG_COMMENTS": "Initial config with COMPANIES",
    "SYS_CREATE_DT": "2025-10-17T23:20:48Z"
},
{
    "CONFIG_ID": 4059713681,
    "CONFIG_COMMENTS": "Data Sources: WATCHLIST",
    "SYS_CREATE_DT": "2025-10-17T23:20:48Z"
},
{
    "CONFIG_ID": 4209002856,
    "CONFIG_COMMENTS": "Added CUSTOMERS data source",
    "SYS_CREATE_DT": "2025-10-17T23:20:48Z"
}
]
}

```

Remarks

The registry contains the original timestamp, original comment and configuration ID of all configurations ever registered with the repository.

NOTE: Registered configurations cannot be unregistered.

Exceptions

[SzException](#)

If a failure occurs.

GetDefaultConfigID()

Gets the default configuration ID for the repository.

```
long GetDefaultConfigID()
```

Returns

[long](#)

The current default configuration ID, or zero (0) if the default configuration has not been set.

Examples

Usage:

```
// How to get the registered default configuration ID
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the config manager
    SzConfigManager configMgr = env.GetConfigManager();

    // get the default configuration ID
    long configID = configMgr.GetDefaultConfigID();

    // check if no default configuration ID is registered
    if (configID == 0)
    {
        // handle having no registered configuration ID
        . . .

    }
    else
    {
        // do something with the configuration ID
    }
}
```

```
    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to get the default configuration ID.", e);
}
```

Remarks

Unless an explicit configuration ID is specified at initialization, the default configuration ID is used.

NOTE: The default configuration ID may not be the same as the active configuration ID.

Exceptions

[SzException](#)

If a failure occurs.

See Also

[Code Snippet: Register Data Sources](#) ↗

RegisterConfig(string)

Registers a configuration definition in the repository with an auto-generated comment.

```
long RegisterConfig(string configDefinition)
```

Parameters

`configDefinition` [string](#) ↗

The configuration definition to register.

Returns

[long](#) ↗

The identifier for referencing the config in the entity repository.

Examples

Usage:

```
// How to register a configuration with the Senzing repository
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the config manager
    SzConfigManager configMgr = env.GetConfigManager();

    // obtain a JSON config definition (will vary by application)
    string configDefinition = CreateConfigWithDataSources("EMPLOYEES");

    // register the config (using an auto-generated comment)
    long configID = configMgr.RegisterConfig(configDefinition);

    // do something with the config ID
    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to register configuration.", e);
}
```

Remarks

NOTE: Registered configurations do not become immediately active nor do they become the default. Further, registered configurations cannot be unregistered.

Exceptions

[SzException](#)

If a failure occurs.

See Also

[Code Snippet: Register Data Sources](#) ↗

RegisterConfig(string, string)

Registers a configuration definition in the repository.

```
long RegisterConfig(string configDefinition, string configComment)
```

Parameters

configDefinition [string](#)

The configuration definition to register.

configComment [string](#)

The comments for the configuration.

Returns

[long](#)

The identifier for referencing the config in the entity repository.

Examples

Usage:

```
// How to register a configuration with the Senzing repository
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the config manager
    SzConfigManager configMgr = env.GetConfigManager();

    // obtain a JSON config definition (will vary by application)
    string configDefinition = CreateConfigWithDataSources("CUSTOMERS");

    // register the config with a custom comment
    long configID = configMgr.RegisterConfig(configDefinition, "Added CUSTOMERS
data source");

    // do something with the config ID
    ...
}
```

```
}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to register configuration.", e);
}
```

Remarks

NOTE: Registered configurations do not become immediately active nor do they become the default. Further, registered configurations cannot be unregistered.

Exceptions

[SzException](#)

If a failure occurs.

See Also

[Code Snippet: Register Data Sources](#)

ReplaceDefaultConfigID(long, long)

Replaces the existing default configuration ID with a new configuration ID.

```
void ReplaceDefaultConfigID(long currentDefaultConfigID, long newDefaultConfigID)
```

Parameters

[currentDefaultConfigID](#) [long](#)

The configuration ID that is believed to be the current default configuration ID.

[newDefaultConfigID](#) [long](#)

The new configuration ID for the repository.

Examples

Usage:

```

// How to replace the registered default configuration ID
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the config manager
    SzConfigManager configMgr = env.GetConfigManager();

    do
    {
        // get the current default configuration ID
        long oldConfigID = configMgr.GetDefaultConfigID();

        // Create a new config (usually modifying the current -- varies
        // by application)
        string configDefinition = AddDataSourcesToConfig(oldConfigID, "PASSENGERS");
        long newConfigID = configMgr.RegisterConfig(configDefinition, "Added
PASSENGERS data source");

        try
        {
            // replace the default config ID with the new config ID
            configMgr.ReplaceDefaultConfigID(oldConfigID, newConfigID);

            // if we get here then break out of the loop
            break;
        }

        catch (SzReplaceConflictException)
        {
            // race condition detected
            // do nothing so we loop through and try again
        }
    }

    } while (true);

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to replace default configuration ID.", e);
}

```

Remarks

The change is prevented (with an [SzReplaceConflictException](#) being thrown) if the current default configuration ID value is not as expected. Use this in place of [SetDefaultConfigID\(long\)](#) to handle race conditions.

Exceptions

[SzReplaceConflictException](#)

If the default configuration ID was not updated to the specified new value because the current default configuration ID found in the repository was not equal to the specified expected current default configuration ID value.

[SzException](#)

If a failure occurs.

See Also

[Code Snippet: Register Data Sources](#)

SetDefaultConfig(string)

Registers a configuration in the repository and then sets its ID as the default for the repository with an auto-generated comment.

```
long SetDefaultConfig(string configDefinition)
```

Parameters

`configDefinition` [string](#)

The configuration definition to register as the default.

Returns

[long](#)

The identifier for referencing the config in the entity repository.

Examples

Usage:

```
// How to set the registered default configuration ID
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the config manager
    SzConfigManager configMgr = env.GetConfigManager();

    // get the configuration ID (varies by application)
    string configDefinition = CreateConfigWithDataSources("VIPS");

    // set the default config (using an auto-generated comment)
    long configID = configMgr.SetDefaultConfig(configDefinition);

    // do something with the registered config ID
    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to set default configuration.", e);
}
```

Remarks

This is a convenience method for [RegisterConfig\(string\)](#) followed by [SetDefaultConfigID\(long\)](#).

Exceptions

[SzException](#)

If a failure occurs.

See Also

[Code Snippet: Initialize Config](#) ↗

[SetDefaultConfig\(string, string\)](#)

Registers a configuration in the repository and then sets its ID as the default for the repository.

```
long SetDefaultConfig(string configDefinition, string configComment)
```

Parameters

configDefinition [string](#)

The configuration definition to register as the default.

configComment [string](#)

The comments for the configuration.

Returns

[long](#)

The identifier for referencing the config in the entity repository.

Examples

Usage:

```
// How to set the registered default configuration ID
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the config manager
    SzConfigManager configMgr = env.GetConfigManager();

    // get the configuration ID (varies by application)
    string configDefinition = CreateConfigWithDataSources("COMPANIES");

    // set the default config (using a specific comment)
    long configID = configMgr.SetDefaultConfig(configDefinition, "Initial config
with COMPANIES");

    // do something with the registered config ID
    . . .
}
```

```
    }
    catch (SzException e)
    {
        // handle or rethrow the exception
        LogError("Failed to set default configuration.", e);
    }
}
```

Remarks

This is a convenience method for [RegisterConfig\(string, string\)](#) followed by [SetDefaultConfigID\(long\)](#).

Exceptions

[SzException](#)

If a failure occurs.

See Also

[Code Snippet: Initialize Config](#) ↗

SetDefaultConfigID(long)

Sets the default configuration ID.

```
void SetDefaultConfigID(long configID)
```

Parameters

configID [long](#) ↗

The configuration ID to set as the default configuration.

Examples

Usage:

```
// How to set the registered default configuration ID
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();
```

```
// get the config manager
SzConfigManager configMgr = env.GetConfigManager();

// get the configuration ID (varies by application)
string configDefinition = CreateConfigWithDataSources("WATCHLIST");
long configID = configMgr.RegisterConfig(configDefinition);

// set the default config ID
configMgr.SetDefaultConfigID(configID);

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to set default configuration ID.", e);
}
```

Remarks

Usually this method is sufficient for setting the default configuration ID. However in concurrent environments that could encounter race conditions, consider using [ReplaceDefaultConfigID\(long, long\)](#) instead.

Exceptions

[SzException](#)

If a failure occurs.

See Also

[Code Snippet: Initialize Config](#) ↗

Class SzConfigRetryable

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Used to annotate which Senzing SDK methods are dependent upon the active configuration being current.

```
[AttributeUsage(AttributeTargets.Method)]
public class SzConfigRetryable : Attribute
```

Inheritance

[object](#) ← [Attribute](#) ← SzConfigRetryable

Inherited Members

[Attribute.Equals\(object\)](#) , [Attribute.GetCustomAttribute\(Assembly, Type\)](#) ,
[Attribute.GetCustomAttribute\(Assembly, Type, bool\)](#) ,
[Attribute.GetCustomAttribute\(MemberInfo, Type\)](#) ,
[Attribute.GetCustomAttribute\(MemberInfo, Type, bool\)](#) ,
[Attribute.GetCustomAttribute\(Module, Type\)](#) ,
[Attribute.GetCustomAttribute\(Module, Type, bool\)](#) ,
[Attribute.GetCustomAttribute\(ParameterInfo, Type\)](#) ,
[Attribute.GetCustomAttribute\(ParameterInfo, Type, bool\)](#) ,
[Attribute.GetCustomAttributes\(Assembly\)](#) ,
[Attribute.GetCustomAttributes\(Assembly, bool\)](#) ,
[Attribute.GetCustomAttributes\(Assembly, Type\)](#) ,
[Attribute.GetCustomAttributes\(Assembly, Type, bool\)](#) ,
[Attribute.GetCustomAttributes\(MemberInfo\)](#) ,
[Attribute.GetCustomAttributes\(MemberInfo, bool\)](#) ,
[Attribute.GetCustomAttributes\(MemberInfo, Type\)](#) ,
[Attribute.GetCustomAttributes\(MemberInfo, Type, bool\)](#) ,
[Attribute.GetCustomAttributes\(Module\)](#) , [Attribute.GetCustomAttributes\(Module, bool\)](#) ,
[Attribute.GetCustomAttributes\(Module, Type\)](#) ,
[Attribute.GetCustomAttributes\(Module, Type, bool\)](#) ,
[Attribute.GetCustomAttributes\(ParameterInfo\)](#) ,
[Attribute.GetCustomAttributes\(ParameterInfo, bool\)](#) ,
[Attribute.GetCustomAttributes\(ParameterInfo, Type\)](#) ,
[Attribute.GetCustomAttributes\(ParameterInfo, Type, bool\)](#) , [Attribute.GetHashCode\(\)](#) ,
[Attribute.IsDefaultAttribute\(\)](#) , [Attribute.IsDefined\(Assembly, Type\)](#) ,

[Attribute.IsDefined\(Assembly, Type, bool\)](#) , [Attribute.IsDefined\(MemberInfo, Type\)](#) ,
[Attribute.IsDefined\(MemberInfo, Type, bool\)](#) , [Attribute.IsDefined\(Module, Type\)](#) ,
[Attribute.IsDefined\(Module, Type, bool\)](#) , [Attribute.IsDefined\(ParameterInfo, Type\)](#) ,
[Attribute.IsDefined\(ParameterInfo, Type, bool\)](#) , [Attribute.Match\(object\)](#) ,
[Attribute.TypeId](#) , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

Methods annotated with this annotation are those methods that may experience a failure if the [active configuration](#) is **not** the most recent configuration for the repository. Such failures can occur when retrieved data references a configuration element (e.g.: data source, feature type or match type) that is not found in the [active configuration](#).

There are various ways this can happen, but typically it means that data has been loaded through another [SzEnvironment](#) using a newer configuration (usually in another system process) and then that data is retrieved by an [SzEnvironment](#) that was initialized prior to that configuration being [registered in the repository](#) and being set as the [default configuration](#).

Such failures can usually be resolved by retrying after checking if the [active configuration ID](#) differs from the current [default configuration ID](#), and if so [reinitializing](#) using the current [default configuration ID](#)

Constructors

SzConfigRetryable()

Default constructor.

```
public SzConfigRetryable()
```

Class SzConfigurationException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Defines an exceptional condition when a failure has occurred pertaining to the Senzing configuration.

```
public class SzConfigurationException : SzException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← SzConfigurationException

Implements

[ISerializable](#)

Inherited Members

[SzException.ErrorCode](#) , [Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

SzConfigurationException()

Default constructor.

```
public SzConfigurationException()
```

SzConfigurationException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzConfigurationException(Exception cause)
```

Parameters

cause [Exception](#)

The underlying cause for the exception.

SzConfigurationException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzConfigurationException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzConfigurationException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzConfigurationException(long? errorCode, string message, Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzConfigurationException(string)

Constructs with a message explaining the reason for the exception.

```
public SzConfigurationException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzConfigurationException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzConfigurationException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Class SzDatabaseConnectionLostException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Extends [SzRetryableException](#) to define an exceptional condition where a database connection was lost causing a Senzing operation to fail. Retrying the operation would likely result in the connection being reestablished and the operation succeeding.

```
public class SzDatabaseConnectionLostException : SzRetryableException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← [SzRetryableException](#) ← SzDatabaseConnectionLostException

Implements

[ISerializable](#)

Inherited Members

[SzException.ErrorCode](#) , [Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

SzDatabaseConnectionLostException()

Default constructor.

```
public SzDatabaseConnectionLostException()
```

SzDatabaseConnectionLostException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzDatabaseConnectionLostException(Exception cause)
```

Parameters

cause [Exception](#)

The underlying cause for the exception.

SzDatabaseConnectionLostException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzDatabaseConnectionLostException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzDatabaseConnectionLostException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzDatabaseConnectionLostException(long? errorCode, string message, Exception cause)
```

Parameters

errorCode [long](#)?

message [string](#)

cause [Exception](#)

The underlying cause for the exception.

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzDatabaseConnectionLostException(string)

Constructs with a message explaining the reason for the exception.

```
public SzDatabaseConnectionLostException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzDatabaseConnectionLostException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzDatabaseConnectionLostException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Class SzDatabaseException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Extends [SzUnrecoverableException](#) to define an exceptional condition triggered by a database error from which we cannot recover (e.g.: missing or unexpected schema definition).

```
public class SzDatabaseException : SzUnrecoverableException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← [SzUnrecoverableException](#) ← [SzDatabaseException](#)

Implements

[ISerializable](#)

Inherited Members

[SzException.ErrorCode](#) , [Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

SzDatabaseException()

Default constructor.

```
public SzDatabaseException()
```

SzDatabaseException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzDatabaseException(Exception cause)
```

Parameters

cause [Exception](#)

The underlying cause for the exception.

SzDatabaseException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzDatabaseException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzDatabaseException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzDatabaseException(long? errorCode, string message, Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzDatabaseException(string)

Constructs with a message explaining the reason for the exception.

```
public SzDatabaseException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzDatabaseException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzDatabaseException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Class SzDatabaseTransientException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Extends [SzRetryableException](#) to define an exceptional condition where an operation failed because a database condition that is transient and would like be resolved on a repeated attempt. Retrying the operation may result in it completing successfully.

```
public class SzDatabaseTransientException : SzRetryableException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← [SzRetryableException](#) ← SzDatabaseTransientException

Implements

[ISerializable](#)

Inherited Members

[SzException.ErrorCode](#) , [Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

SzDatabaseTransientException()

Default constructor.

```
public SzDatabaseTransientException()
```

SzDatabaseTransientException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzDatabaseTransientException(Exception cause)
```

Parameters

cause [Exception](#)

The underlying cause for the exception.

SzDatabaseTransientException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzDatabaseTransientException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzDatabaseTransientException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzDatabaseTransientException(long? errorCode, string message,  
Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzDatabaseTransientException(string)

Constructs with a message explaining the reason for the exception.

```
public SzDatabaseTransientException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzDatabaseTransientException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzDatabaseTransientException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Interface SzDiagnostic

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Defines the interface to the Senzing diagnostic functions.

```
public interface SzDiagnostic
```

Examples

An `SzDiagnostic` instance is typically obtained from an [SzEnvironment](#) instance via the [GetDiagnostic\(\)](#) method as follows.

For example:

```
// How to obtain an SzDiagnostic instance
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    SzDiagnostic diagnostic = env.GetDiagnostic();

    . . .

}
catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to get SzDiagnostic.", e);
}
```

Remarks

The Senzing diagnostic functions provide diagnostics and statistics pertaining to the host system and the Senzing repository.

Methods

CheckRepositoryPerformance(int)

Conducts a rudimentary repository test to gauge I/O and network performance.

```
string CheckRepositoryPerformance(int secondsToRun)
```

Parameters

secondsToRun [int](#)

How long to run the database performance test.

Returns

[string](#)

The JSON [string](#) describing the results of the performance test.

Examples

Usage:

```
// How to get repository info via SzDiagnostic
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the diagnostic instance
    SzDiagnostic diagnostic = env.GetDiagnostic();

    // check the repository performance
    string result = diagnostic.CheckRepositoryPerformance(10);
    demoResult = result; // @replace
    // do something with the returned JSON (varies by application)
    Log(result);

}
catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to check the repository performance.", e);
}
```

Example Result: (formatted for readability)

```
{  
    "numRecordsInserted": 337000,  
    "insertTime": 10000  
}
```

Remarks

Typically, this is only run when troubleshooting performance. This is a non-destructive test.

Exceptions

[SzException](#)

Thrown if a failure occurs.

GetFeature(long)

Experimental/internal for Senzing support use.

```
[SzConfigRetryable]  
string GetFeature(long featureID)
```

Parameters

[featureID long](#)

The identifier for the feature.

Returns

[string](#)

The feature definition describing the feature for the specified feature ID.

Examples

Usage:

```

// How to get a feature by its feature ID
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the diagnostic instance
    SzDiagnostic diagnostic = env.GetDiagnostic();

    // get a valid feature (varies by application)
    long featureID = GetFeatureID();

    // get the feature for the feature ID
    string result = diagnostic.GetFeature(featureID);
    demoResult = result; // @replace
    // do something with the returned JSON
    Log(result);

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to purge the repository.", e);
}

```

Example Result: (formatted for readability)

```
{
    "LIB_FEAT_ID": 6,
    "FTYPE_CODE": "PHONE_KEY",
    "ELEMENTS": [
        {
            "FELEM_CODE": "EXPRESSION",
            "FELEM_VALUE": "7025551212"
        }
    ]
}
```

Exceptions

[SzException](#)

Thrown if a failure occurs.

GetRepositoryInfo()

Returns overview information about the repository.

```
string GetRepositoryInfo()
```

Returns

[string](#)

A JSON string describing the datastore.

Examples

Usage:

```
// How to get repository info via SzDiagnostic
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the diagnostic instance
    SzDiagnostic diagnostic = env.GetDiagnostic();

    // get the repository info
    string info = diagnostic.GetRepositoryInfo();
    demoResult = info; // @replace
    // do something with the returned JSON (varies by application)
    Log(info);

}
catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to get the repository info.", e);
}
```

Example Result: (formatted for readability)

```
{
  "dataStores": [
    {
```

```
        "id": "CORE",
        "type": "sqlite3",
        "location": "/tmp/sz-example-4f518d0b-780a-44f6-a940-ab98515cee08/G2C.db"
    }
]
}
```

Exceptions

SzException

Thrown if a failure occurs.

PurgeRepository()

Permanently deletes all data in the repository, except the configuration.

```
void PurgeRepository()
```

Examples

Usage:

```
// How to purge the Senzing repository
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the diagnostic instance
    SzDiagnostic diagnostic = env.GetDiagnostic();

    // purge the repository (MAKE SURE YOU WANT TO DO THIS)
    if (ConfirmPurge())
    {
        diagnostic.PurgeRepository();
    }

}
catch (SzException e)
{
    // handle or rethrow the exception
}
```

```
        LogError("Failed to purge the repository.", e);  
    }
```

Remarks

WARNING: This method is destructive, it will delete all loaded records and entity resolution decisions. Senzing does not provide a means to restore the data. The only means of recovery would be restoring from a database backup.

Exceptions

[SzException](#)

Thrown if a failure occurs.

See Also

[Purge Repository Code Snippet](#)

Interface SzEngine

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Defines the interface to the Senzing engine functions.

```
public interface SzEngine
```

Examples

An `SzEngine` instance is typically obtained from an `SzEnvironment` instance via the [GetEngine\(\)](#) method as follows.

For example:

```
// How to obtain an SzEngine instance
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to get SzEngine.", e);
}
```

Remarks

The Senzing engine functions primarily provide means of working with identity data records, entities and their relationships.

Methods

AddRecord(string, string, string, SzFlag?)

Loads a record into the repository and performs entity resolution.

```
[SzConfigRetryable]  
string AddRecord(string dataSourceCode, string recordID, string recordDefinition,  
SzFlag? flags = (SzFlag)0)
```

Parameters

dataSourceCode [string](#)

The data source code identifying the data source for the record being added.

recordID [string](#)

The record ID that uniquely identifies the record being added within the scope of its associated data source.

recordDefinition [string](#)

The [string](#) that defines the record, typically in JSON format.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzAddRecordFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter will default its value to [SzAddRecordDefaultFlags](#). Specify [SzWithInfo](#) for an INFO response. Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) result produced by adding the record to the repository, or [null](#) if the specified flags do not indicate that an INFO message should be returned.

Examples

Usage:

```
// How to load a record  
try
```

```

{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get a record definition (varies by application)
    string recordDefinition =
        """
        {
            "DATA_SOURCE": "TEST",
            "RECORD_ID": "ABC123",
            "NAME_FULL": "Joe Schmoe",
            "PHONE_NUMBER": "702-555-1212",
            "EMAIL_ADDRESS": "joeschmoe@nowhere.com"
        }
        """;
}

// add the record to the repository
string info = engine.AddRecord(
    "TEST", "ABC123", recordDefinition, SzWithInfo);

// do something with the "info JSON" (varies by application)
JsonObject? jsonObject = JsonNode.Parse(info)?.AsObject();
if (jsonObject != null && jsonObject.ContainsKey("AFFECTED_ENTITIES"))
{
    JSONArray? affectedArr = jsonObject["AFFECTED_ENTITIES"]?.AsArray();
    for (int index = 0; affectedArr != null && index <
affectedArr.Count; index++)
    {
        JsonObject? affected = affectedArr[index]?.AsObject();
        long affectedID = affected?["ENTITY_ID"]?.GetValue<long>() ?? 0L;

        . . .

    }
}

}

catch (SzUnknownDataSourceException e)
{
    // handle the unknown data source exception
    LogError("Expected data source is not configured.", e);
}

catch (SzException e)

```

```
{  
    // handle or rethrow the exception  
    LogError("Failed to add record.", e);  
}
```

Example Result: (formatted for readability)

```
{  
    "DATA_SOURCE": "TEST",  
    "RECORD_ID": "ABC123",  
    "AFFECTED_ENTITIES": [  
        {  
            "ENTITY_ID": 100002  
        }  
    ]  
}
```

Remarks

If a record already exists with the same data source code and record ID, it will be replaced. If the record definition contains `DATA_SOURCE` and `RECORD_ID` JSON keys, the values must match the `dataSourceCode` and `recordId` parameters.

The optionally specified bitwise-OR'd [SzFlag](#) values not only controls how the operation is performed, but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzAddRecordFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzUnknownDataSourceException](#)

If an unrecognized data source code is specified.

[SzBadInputException](#)

If the specified record definition has a data source or record ID value that conflicts with the specified data source code and/or record ID values.

[SzException](#)

If a failure occurs.

See Also

[SzNoFlags](#), [SzWithInfo](#), [SzAddRecordDefaultFlags](#), [SzAddRecordFlags](#),
[Code Snippet: Load Records](#), [Code Snippet: Load Truth Set "With Info"](#),
[Code Snippet: Load via Futures](#), [Code Snippet: Load via Loop](#),
[Code Snippet: Load via Queue](#), [Code Snippet: Load "With Info" via Futures](#),
[Code Snippet: Load "With Stats" Via Loop](#)

CloseExportReport(IntPtr)

Closes an export report.

```
void CloseExportReport(IntPtr exportHandle)
```

Parameters

exportHandle [IntPtr](#)

The export handle of the export report to close.

Examples

Usage (JSON Export):

```
// How to export entity data in JSON format
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // export the JSON data
    IntPtr exportHandle = engine.ExportJsonEntityReport(SzExportDefaultFlags);

    // read the data
    try
    {
        // fetch the first JSON record
        string jsonData = engine.FetchNext(exportHandle);

        while (jsonData != null)
        {
```

```

        // parse the JSON data
        JsonObject? jsonObject = JsonNode.Parse(jsonData)?.AsObject();

        // do something with the parsed data (varies by application)
        ProcessJsonRecord(jsonObject);

        // fetch the next JSON record
        jsonData = engine.FetchNext(exportHandle);
    }

}

finally
{
    // close the export handle
    engine.CloseExportReport(exportHandle);
}

}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to perform JSON export.", e);
}

```

Usage (CSV Export):

```

// How to export entity data in CSV format
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // export the JSON data
    IntPtr exportHandle = engine.ExportCsvEntityReport("*.entity", SzExportDefaultFlags);

    // read the data
    try
    {
        // fetch the CSV header line from the exported data
        string csvHeaders = engine.FetchNext(exportHandle);

        // process the CSV headers (varies by application)
    }
}

```

```

ProcessCsvHeaders(csvHeaders);

// fetch the first CSV record from the exported data
string csvRecord = engine.FetchNext(exportHandle);

while (csvRecord != null)
{
    // do something with the exported record (varies by application)
    ProcessCsvRecord(csvRecord);

    // fetch the next exported CSV record
    csvRecord = engine.FetchNext(exportHandle);
}

}

finally
{
    // close the export handle
    engine.CloseExportReport(exportHandle);
}

}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to perform CSV export.", e);
}

```

Remarks

Used in conjunction with [ExportJsonEntityReport\(SzFlag?\)](#), [ExportCsvEntityReport\(string, SzFlag?\)](#) and [FetchNext\(IntPtr\)](#).

Exceptions

[SzException](#)

If the specified export handle has already been [closed](#) or if any other failure occurs.

See Also

[FetchNext\(IntPtr\)](#), [ExportJsonEntityReport\(SzFlag?\)](#), [ExportCsvEntityReport\(string, SzFlag?\)](#)

CountRedoRecords()

Gets the number of redo records pending processing.

```
long CountredoRecords()
```

Returns

[long](#)

The number of redo records pending to be processed.

Examples

Usage:

```
// How to check for and process redo records
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // loop through any redo records
    for (string redoRecord = engine.GetredoRecord();
        redoRecord != null;
        redoRecord = engine.GetredoRecord())
    {
        try
        {
            // process the redo record
            string info = engine.ProcessredoRecord(redoRecord, SzWithInfo);

            // do something with the "info JSON" (varies by application)
            JsonObject? jsonObject = JsonNode.Parse(info)?.AsObject();
            if (jsonObject != null && jsonObject.ContainsKey("AFFECTED_ENTITIES"))
            {
                JSONArray? affectedArr = jsonObject["AFFECTED_ENTITIES"]?.AsArray();
                for (int index = 0; affectedArr != null && index <
affectedArr.Count; index++)
                {
                    JsonObject? affected = affectedArr[index]?.AsObject();
                    long affectedID = affected?["ENTITY_ID"]?.GetValue<long>()
?? 0L;
                }
            }
        }
    }
}
```

```

        . . .
    }

}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to process redo record: " + redoRecord, e);
}
}

// get the redo count
long redoCount = engine.CountRedoRecords();

// do something with the redo count
Log("Pending Redos: " + redoCount);
}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to process redos.", e);
}

```

Remarks

WARNING: When there is a large number of redo records, this is an expensive call. Hint: If processing redo records, use result of [GetredoRecord\(\)](#) to manage looping.

This method is used in conjunction with [GetredoRecord\(\)](#) and [ProcessredoRecord\(string, SzFlag?\)](#).

Exceptions

[SzException](#)

If a failure occurs.

See Also

[ProcessredoRecord\(string, SzFlag?\)](#), [GetredoRecord\(\)](#),

[Code Snippet: Processing Redos while Loading](#) ↴,

[Code Snippet: Continuous Redo Processing](#) ↴,

[Code Snippet: Continuous Redo Processing via Futures](#),

[Code Snippet: Continuous Redo "With Info" Processing](#)

DeleteRecord(string, string, SzFlag?)

Deletes a record from the repository and performs entity resolution.

```
[SzConfigRetryable]
string DeleteRecord(string dataSourceCode, string recordID, SzFlag? flags
= (SzFlag)0)
```

Parameters

dataSourceCode [string](#)

The data source code identifying the data source for the record being deleted.

recordID [string](#)

The record ID that uniquely identifies the record being deleted within the scope of its associated data source.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzDeleteRecordFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter will default its value to [SzDeleteRecordDefaultFlags](#). Specify [SzWithInfo](#) for an INFO response. Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) result produced by deleting the record from the repository, or [null](#) if the specified flags do not indicate that an INFO message should be returned.

Examples

Usage:

```
// How to delete a record
try
```

```

{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // delete the record from the repository
    string info = engine.DeleteRecord("TEST", "ABC123", SzWithInfo);

    // do something with the "info JSON" (varies by application)
    JsonObject? jsonObject = JsonNode.Parse(info)?.AsObject();
    if (jsonObject != null && jsonObject.ContainsKey("AFFECTED_ENTITIES"))
    {
        JSONArray? affectedArr = jsonObject["AFFECTED_ENTITIES"]?.AsArray();

        for (int index = 0; affectedArr != null && index <
affectedArr.Count; index++)
        {
            JsonObject? affected = affectedArr[index]?.AsObject();

            long affectedID = affected?["ENTITY_ID"]?.GetValue<long>() ?? 0L;

            . . .
        }
    }
}

}

catch (SzUnknownDataSourceException e)
{
    // handle the unknown data source exception
    LogError("Expected data source is not configured.", e);
}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to delete record.", e);
}

```

Example Result:

(formatted for readability)

```
{
    "DATA_SOURCE": "TEST",
    "RECORD_ID": "ABC123",
}
```

```
"AFFECTED_ENTITIES": [
  {
    "ENTITY_ID": 100001
  }
]
}
```

Remarks

NOTE: This method is idempotent in that it succeeds with no changes being made when the record is not found in the repository.

The optionally specified bitwise-OR'd [SzFlag](#) values not only controls how the operation is performed, but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzDeleteRecordFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzUnknownDataSourceException](#)

If an unrecognized data source code is specified.

[SzException](#)

If a failure occurs.

See Also

[SzNoFlags](#), [SzWithInfo](#), [SzDeleteRecordFlags](#), [SzDeleteRecordDefaultFlags](#),
[Code Snippet: Delete via Loop](#), [Code Snippet: Delete via Futures](#),
[Code Snippet: Delete "With Info" via Futures](#)

ExportCsvEntityReport(string, SzFlag?)

Initiates an export report of entity data in CSV format.

```
[SzConfigRetryable]
IntPtr ExportCsvEntityReport(string csvColumnList, SzFlag? flags =
SzFlag.SzExportIncludeMultiRecordEntities |
SzFlag.SzExportIncludeSingleRecordEntities |
SzFlag.SzEntityIncludePossiblySameRelations |
SzFlag.SzEntityIncludePossiblyRelatedRelations |
SzFlag.SzEntityIncludeNameOnlyRelations | SzFlag.SzEntityIncludeDisclosedRelations |
```

```
SzFlag.SzEntityIncludeRepresentativeFeatures | SzFlag.SzEntityIncludeEntityName |
SzFlag.SzEntityIncludeRecordSummary | SzFlag.SzEntityIncludeRecordData |
SzFlag.SzEntityIncludeRecordMatchingInfo | SzFlag.SzEntityIncludeRelatedEntityName |
SzFlag.SzEntityIncludeRelatedMatchingInfo |
SzFlag.SzEntityIncludeRelatedRecordSummary)
```

Parameters

csvColumnList [string](#)

Specify "*" to indicate "all columns", specify empty-string to indicate the "standard columns", otherwise specify a comma-separated list of column names.

flags [SzFlag](#)?

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzExportFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter defaults it value to [SzExportDefaultFlags](#) for the default recommended flags. Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[IntPtr](#)

The export handle to use for retrieving the export data.

Examples

Usage:

```
// How to export entity data in CSV format
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // export the JSON data
    IntPtr exportHandle = engine.ExportCsvEntityReport("*", SzExportDefaultFlags);

    // read the data
    try
```

```

{
    // fetch the CSV header line from the exported data
    string csvHeaders = engine.FetchNext(exportHandle);

    // process the CSV headers (varies by application)
    ProcessCsvHeaders(csvHeaders);

    // fetch the first CSV record from the exported data
    string csvRecord = engine.FetchNext(exportHandle);

    while (csvRecord != null)
    {
        // do something with the exported record (varies by application)
        ProcessCsvRecord(csvRecord);

        // fetch the next exported CSV record
        csvRecord = engine.FetchNext(exportHandle);
    }

}

finally
{
    // close the export handle
    engine.CloseExportReport(exportHandle);
}

}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to perform CSV export.", e);
}

```

Example Complete Export Report:

```

RESOLVED_ENTITY_ID,RESOLVED_ENTITY_NAME,RELATED_ENTITY_ID,MATCH_LEVEL,MATCH_LEVEL_CO
DE,MATCH_KEY,MATCH_KEY_DETAILS,IS_DISCLOSED,IS_AMBIGUOUS,DATA_SOURCE,RECORD_ID,JSON_
DATA,FIRST_SEEN_DT,LAST_SEEN_DT,UNMAPPED_DATA,ERRULE_CODE,RELATED_ENTITY_NAME
1,"Joseph Schmidt",0,0,"","","","",0,0,"PASSENGERS","ABC123",""
{""RECORD_ID"":""ABC123"" , ""NAME_FIRST"":""Joseph"" , ""NAME_LAST"":""Schmidt"" , ""MOBI
LE_PHONE_NUMBER"":""213-555-1212"" , ""HOME_PHONE_NUMBER"":""818-777-
2424"" , ""ADDR_FULL"":""101 Main Street, Los Angeles, CA
90011"" , ""DATE_OF_BIRTH"":""12-JAN-1981"" , ""DATA_SOURCE"":""PASSENGERS""} , "2025-10-
17T23:21:01Z", "2025-10-17T23:21:01Z", "{}", "", ""
1,"Joseph Schmidt",2,3,"POSSIBLY RELATED", "+PHONE-DOB", {"CONFIRMATIONS": "

```

```
[{"TOKEN": "PHONE", "FTYPE_CODE": "PHONE", "SCORE_BEHAVIOR": "F1", "SOURCE": "PHONE", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 5, "INBOUND_FEAT_DESC": "213-555-1212"}, {"INBOUND_FEAT_USAGE_TYPE": "MOBILE", "CANDIDATE_FEAT_ID": 5, "CANDIDATE_FEAT_DESC": "213-555-1212"}, {"CANDIDATE_FEAT_USAGE_TYPE": "MOBILE", "SCORE_BUCKET": "SAME"}], "DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 79, "ADDITIONAL_SCORES": [{"FULL_SCORE": 79}, {"INBOUND_FEAT_ID": 21, "INBOUND_FEAT_DESC": "15-MAR-1982"}, {"CANDIDATE_FEAT_ID": 2, "CANDIDATE_FEAT_DESC": "12-JAN-1981"}, {"SCORE_BUCKET": "NO_CHANCE"}]}, {"RECORD_ID": "DEF456", "NAME_FIRST": "Joann", "NAME_LAST": "Smith", "MOBILE_PHONE_NUMBER": "213-555-1212", "HOME_PHONE_NUMBER": "818-888-3939", "ADDR_FULL": "101 Fifth Ave, Los Angeles, CA 90018", "DATE_OF_BIRTH": "15-MAR-1982", "DATA_SOURCE": "PASSENGERS"}, {"RECORD_ID": "DEF456", "NAME_FIRST": "Joseph Schmidt", "NAME_LAST": "Schmidt", "MOBILE_PHONE_NUMBER": "213-555-1212", "HOME_PHONE_NUMBER": "818-888-3939", "ADDR_FULL": "101 Main Street, Los Angeles, CA 90011", "DATE_OF_BIRTH": "15-MAR-1982", "DATA_SOURCE": "PASSENGERS"}, {"RECORD_ID": "MN0345", "NAME_FIRST": "Bill", "NAME_LAST": "Bandley", "MOBILE_PHONE_NUMBER": "818-444-2121", "HOME_PHONE_NUMBER": "818-123-4567", "ADDR_FULL": "101 Main Street, Los Angeles, CA 90011", "DATE_OF_BIRTH": "22-AUG-1981", "DATA_SOURCE": "EMPLOYEES"}, {"RECORD_ID": "DEF456", "NAME_FIRST": "Joann", "NAME_LAST": "Smith", "MOBILE_PHONE_NUMBER": "213-555-1212", "HOME_PHONE_NUMBER": "818-888-3939", "ADDR_FULL": "101 Fifth Ave, Los Angeles, CA 90018", "DATE_OF_BIRTH": "15-MAR-1982", "DATA_SOURCE": "PASSENGERS"}, {"RECORD_ID": "DEF456", "NAME_FIRST": "Joann", "NAME_LAST": "Smith", "MOBILE_PHONE_NUMBER": "213-555-1212", "HOME_PHONE_NUMBER": "818-888-3939", "ADDR_FULL": "101 Fifth Ave, Los Angeles, CA 90018", "DATE_OF_BIRTH": "15-MAR-1982", "DATA_SOURCE": "PASSENGERS"}], "DENIALS": [{"TOKEN": "PHONE", "FTYPE_CODE": "PHONE", "SCORE_BEHAVIOR": "F1", "SOURCE": "PHONE", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 5, "INBOUND_FEAT_DESC": "213-555-1212"}, {"INBOUND_FEAT_USAGE_TYPE": "MOBILE", "CANDIDATE_FEAT_ID": 5, "CANDIDATE_FEAT_DESC": "213-555-1212"}, {"CANDIDATE_FEAT_USAGE_TYPE": "MOBILE", "SCORE_BUCKET": "SAME"}]}]
```

S"":

[{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 79, "ADDITIONAL_SCORES": [{"FULL_SCORE": 79}, {"INBOUND_FEAT_ID": 2, "INBOUND_FEAT_DESC": "12-JAN-1981"}, {"CANDIDATE_FEAT_ID": 21, "CANDIDATE_FEAT_DESC": "15-MAR-1982"}, {"SCORE_BUCKET": "NO_CHANCE"}], 0, 0, "PASSENGERS", "ABC123", {"RECORD_ID": "ABC123", "NAME_FIRST": "Joseph", "NAME_LAST": "Schmidt", "MOBILE_PHONE_NUMBER": "213-555-1212", "HOME_PHONE_NUMBER": "818-777-2424", "ADDR_FULL": "101 Main Street, Los Angeles, CA 90011", "DATE_OF_BIRTH": "12-JAN-1981", "DATA_SOURCE": "PASSENGERS"}, "2025-10-17T23:21:01Z", "2025-10-17T23:21:01Z", {}, "SF1", "Joseph Schmidt", 2, "Joann Smith", 3, 3, "POSSIBLY RELATED", "+ADDRESS-DOB", {"CONFIRMATIONS": [{"TOKEN": "ADDRESS", "FTYPE_CODE": "ADDRESS", "SCORE_BEHAVIOR": "FF", "SOURCE": "ADDRESS", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 22, "INBOUND_FEAT_DESC": "101 Fifth Ave, Los Angeles, CA 90018"}, {"CANDIDATE_FEAT_ID": 22, "CANDIDATE_FEAT_DESC": "101 Fifth Ave, Los Angeles, CA 90018"}, {"SCORE_BUCKET": "SAME"}]}, {"DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 65, "ADDITIONAL_SCORES": [{"FULL_SCORE": 65}, {"INBOUND_FEAT_ID": 48, "INBOUND_FEAT_DESC": "17-DEC-1977"}, {"CANDIDATE_FEAT_ID": 21, "CANDIDATE_FEAT_DESC": "15-MAR-1982"}, {"SCORE_BUCKET": "NO_CHANCE"}], 0, 0, "PASSENGERS", "GHI789", {"RECORD_ID": "GHI789", "NAME_FIRST": "John", "NAME_LAST": "Parker", "MOBILE_PHONE_NUMBER": "818-555-1313", "HOME_PHONE_NUMBER": "818-999-2121", "ADDR_FULL": "101 Fifth Ave, Los Angeles, CA 90018", "DATE_OF_BIRTH": "17-DEC-1977", "DATA_SOURCE": "PASSENGERS"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "SFF", "John Parker", 2, "Joann Smith", 6, 3, "POSSIBLY RELATED", "+SURNAME+PHONE-DOB", {"CONFIRMATIONS": [{"TOKEN": "SURNAME", "FTYPE_CODE": "NAME", "SCORE_BEHAVIOR": "NAME", "SOURCE": "/SUR_NAME", "SCORE": 67, "ADDITIONAL_SCORES": [{"GNR_FN": 67, "GNR_SN": 100, "GNR_GN": 34, "GENERATION_MATCH": -1, "GNR_ON": -1}, {"INBOUND_FEAT_ID": 106, "INBOUND_FEAT_DESC": "Craig Smith"}, {"CANDIDATE_FEAT_ID": 20, "CANDIDATE_FEAT_DESC": "Joann Smith"}, {"SCORE_BUCKET": "UNLIKELY"}]}, {"TOKEN": "PHONE", "FTYPE_CODE": "PHONE", "SCORE_BEHAVIOR": "FF", "SOURCE": "PHONE", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 23, "INBOUND_FEAT_DESC": "818-888-3939"}, {"INBOUND_FEAT_USAGE_TYPE": "HOME", "CANDIDATE_FEAT_ID": 23, "CANDIDATE_FEAT_DESC": "818-888-3939"}, {"CANDIDATE_FEAT_USAGE_TYPE": "HOME", "SCORE_BUCKET": "SAME"}]}, {"DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 77, "ADDITIONAL_SCORES": [{"FULL_SCORE": 77}, {"INBOUND_FEAT_ID": 107, "INBOUND_FEAT_DESC": "17-OCT-1983"}, {"CANDIDATE_FEAT_ID": 21, "CANDIDATE_FEAT_DESC": "15-MAR-1982"}, {"SCORE_BUCKET": "NO_CHANCE"}], 0, 0, "EMPLOYEES", "PQR678", "}], "EMployees": "PQR678"}]


```

{ "RECORD_ID": "JKL012", "NAME_FIRST": "Jane", "NAME_LAST": "Donaldson", "MOBILE_PHONE_NUMBER": "818-555-1313", "HOME_PHONE_NUMBER": "818-222-3131", "ADDR_FULL": "400 River Street, Pasadena, CA 90034", "DATE_OF_BIRTH": "23-MAY-1973", "DATA_SOURCE": "PASSENGERS"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "", ""
4, "Jane Donaldson", 3, 3, "POSSIBLY RELATED", "+PHONE-DOB", {"CONFIRMATIONS": [{"TOKEN": "PHONE", "FTYPE_CODE": "PHONE", "SCORE_BEHAVIOR": "F1", "SOURCE": "PHONE", "SCORE": 100, "ADDITIONAL_SCORES": {"FULL_SCORE": 100}, "INBOUND_FEAT_ID": 50, "INBOUND_FEAT_DESC": "818-555-1313", "INBOUND_FEAT_USAGE_TYPE": "MOBILE", "CANDIDATE_FEAT_ID": 50, "CANDIDATE_FEAT_DESC": "818-555-1313", "CANDIDATE_FEAT_USAGE_TYPE": "MOBILE", "SCORE_BUCKET": "SAME"}], "DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 63, "ADDITIONAL_SCORES": {"FULL_SCORE": 63}, "INBOUND_FEAT_ID": 48, "INBOUND_FEAT_DESC": "17-DEC-1977", "CANDIDATE_FEAT_ID": 64, "CANDIDATE_FEAT_DESC": "23-MAY-1973", "SCORE_BUCKET": "NO_CHANCE"}]}, 0, 0, "PASSENGERS", "GHI789", "
{ "RECORD_ID": "GHI789", "NAME_FIRST": "John", "NAME_LAST": "Parker", "MOBILE_PHONE_NUMBER": "818-555-1313", "HOME_PHONE_NUMBER": "818-999-2121", "ADDR_FULL": "101 Fifth Ave, Los Angeles, CA 90018", "DATE_OF_BIRTH": "17-DEC-1977", "DATA_SOURCE": "PASSENGERS"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "SF1", "John Parker"
4, "Jane Donaldson", 10, 3, "POSSIBLY RELATED", "+PNAME+ADDRESS-DOB", {"CONFIRMATIONS": [{"TOKEN": "PNAME", "FTYPE_CODE": "NAME", "SCORE_BEHAVIOR": "NAME", "SOURCE": "/PART_NAME", "SCORE": 84, "ADDITIONAL_SCORES": {"GNR_FN": 84, "GNR_SN": 68, "GNR_GN": 100, "GENERATION_MATCH": -1, "GNR_ON": -1}, "INBOUND_FEAT_ID": 195, "INBOUND_FEAT_DESC": "Jane Johnson", "CANDIDATE_FEAT_ID": 63, "CANDIDATE_FEAT_DESC": "Jane Donaldson", "SCORE_BUCKET": "LIKELY"}, {"TOKEN": "ADDRESS", "FTYPE_CODE": "ADDRESS", "SCORE_BEHAVIOR": "FF", "SOURCE": "ADDRESS", "SCORE": 100, "ADDITIONAL_SCORES": {"FULL_SCORE": 100}, "INBOUND_FEAT_ID": 65, "INBOUND_FEAT_DESC": "400 River Street, Pasadena, CA 90034", "CANDIDATE_FEAT_ID": 65, "CANDIDATE_FEAT_DESC": "400 River Street, Pasadena, CA 90034", "SCORE_BUCKET": "SAME"}], "DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 65, "ADDITIONAL_SCORES": {"FULL_SCORE": 65}, "INBOUND_FEAT_ID": 196, "INBOUND_FEAT_DESC": "6-SEP-1975", "CANDIDATE_FEAT_ID": 64, "CANDIDATE_FEAT_DESC": "23-MAY-1973", "SCORE_BUCKET": "NO_CHANCE"}]}, 0, 0, "VIPS", "XYZ234", "
{ "RECORD_ID": "XYZ234", "NAME_FIRST": "Jane", "NAME_LAST": "Johnson", "MOBILE_PHONE_NUMBER": "818-333-7171", "HOME_PHONE_NUMBER": "818-123-9876", "ADDR_FULL": "400 River Street, Pasadena, CA 90034", "DATE_OF_BIRTH": "6-SEP-1975", "DATA_SOURCE": "VIPS"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "SFF", "Jane Johnson"
5, "Bill Bandley", 0, 0, "", "", "", 0, 0, "EMPLOYEES", "MN0345", "

```

```

{ "RECORD_ID": "MN0345", "NAME_FIRST": "Bill", "NAME_LAST": "Bandley", "MOBILE_PHONE_NUMBER": "818-444-2121", "HOME_PHONE_NUMBER": "818-123-4567", "ADDR_FULL": "101 Main Street, Los Angeles, CA 90011", "DATE_OF_BIRTH": "22-AUG-1981", "DATA_SOURCE": "EMPLOYEES"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "", "", 5, "Bill Bandley", 1, 3, "POSSIBLY RELATED", "+ADDRESS-DOB", {"CONFIRMATIONS": [{"TOKEN": "ADDRESS", "FTYPE_CODE": "ADDRESS", "SCORE_BEHAVIOR": "FF", "SOURCE": "ADDRESS", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 3, "INBOUND_FEAT_DESC": "101 Main Street, Los Angeles, CA 90011", "CANDIDATE_FEAT_ID": 3, "CANDIDATE_FEAT_DESC": "101 Main Street, Los Angeles, CA 90011", "SCORE_BUCKET": "SAME"}]}, {"DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 84, "ADDITIONAL_SCORES": [{"FULL_SCORE": 84}, {"INBOUND_FEAT_ID": 2, "INBOUND_FEAT_DESC": "12-JAN-1981", "CANDIDATE_FEAT_ID": 81, "CANDIDATE_FEAT_DESC": "22-AUG-1981", "SCORE_BUCKET": "UNLIKELY"}]}], 0, 0, "PASSENGERS", "ABC123", "RECORD_ID": "ABC123", "NAME_FIRST": "Joseph", "NAME_LAST": "Schmidt", "MOBILE_PHONE_NUMBER": "213-555-1212", "HOME_PHONE_NUMBER": "818-777-2424", "ADDR_FULL": "101 Main Street, Los Angeles, CA 90011", "DATE_OF_BIRTH": "12-JAN-1981", "DATA_SOURCE": "PASSENGERS"}, "2025-10-17T23:21:01Z", "2025-10-17T23:21:01Z", {}, "SFF", "Joseph Schmidt", 5, "Bill Bandley", 8, 3, "POSSIBLY RELATED", "+PHONE-DOB", {"CONFIRMATIONS": [{"TOKEN": "PHONE", "FTYPE_CODE": "PHONE", "SCORE_BEHAVIOR": "F1", "SOURCE": "PHONE", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 83, "INBOUND_FEAT_DESC": "818-444-2121", "INBOUND_FEAT_USAGE_TYPE": "MOBILE", "CANDIDATE_FEAT_ID": 83, "CANDIDATE_FEAT_DESC": "818-444-2121", "CANDIDATE_FEAT_USAGE_TYPE": "MOBILE", "SCORE_BUCKET": "SAME"}]}, {"DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 79, "ADDITIONAL_SCORES": [{"FULL_SCORE": 79}, {"INBOUND_FEAT_ID": 160, "INBOUND_FEAT_DESC": "27-JUN-1980", "CANDIDATE_FEAT_ID": 81, "CANDIDATE_FEAT_DESC": "22-AUG-1981", "SCORE_BUCKET": "NO_CHANCE"}]}], 0, 0, "EMPLOYEES", "DEF890", "RECORD_ID": "DEF890", "NAME_FIRST": "Katrina", "NAME_LAST": "Osmond", "MOBILE_PHONE_NUMBER": "818-444-2121", "HOME_PHONE_NUMBER": "818-111-2222", "ADDR_FULL": "707 Seventh Ave, Los Angeles, CA 90043", "DATE_OF_BIRTH": "27-JUN-1980", "DATA_SOURCE": "EMPLOYEES"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "SF1", "Katrina Osmond", 6, "Craig Smith", 0, 0, "", "", 0, 0, "EMPLOYEES", "PQR678", "RECORD_ID": "PQR678", "NAME_FIRST": "Craig", "NAME_LAST": "Smith", "MOBILE_PHONE_NUMBER": "818-555-1212", "HOME_PHONE_NUMBER": "818-888-3939", "ADDR_FULL": "451 Dover Street, Los Angeles, CA 90018", "DATE_OF_BIRTH": "17-OCT-1983", "DATA_SOURCE": "EMPLOYEES"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "", "", 6, "Craig Smith", 2, 3, "POSSIBLY RELATED", "+SURNAME+PHONE-DOB", {"CONFIRMATIONS": []
}

```

```
[{"TOKEN": "SURNAME", "FTYPE_CODE": "NAME", "SCORE_BEHAVIOR": "NAME", "SOURCE": "/SUR_NAME", "SCORE": 67, "ADDITIONAL_SCORES": [{"GNR_FN": 67, "GNR_SN": 100, "GNR_GN": 34, "GENERATION_MATCH": -1, "GNR_ON": -1}, {"INBOUND_FEAT_ID": 20, "INBOUND_FEAT_DESC": "Joann Smith", "CANDIDATE_FEAT_ID": 106, "CANDIDATE_FEAT_DESC": "Craig Smith", "SCORE_BUCKET": "UNLIKELY"}, {"TOKEN": "PHONE", "FTYPE_CODE": "PHONE", "SCORE_BEHAVIOR": "FF", "SOURCE": "PHONE", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 23, "INBOUND_FEAT_DESC": "818-888-3939", "INBOUND_FEAT_USAGE_TYPE": "HOME", "CANDIDATE_FEAT_ID": 23, "CANDIDATE_FEAT_DESC": "818-888-3939", "CANDIDATE_FEAT_USAGE_TYPE": "HOME", "SCORE_BUCKET": "SAME"}], "DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 77, "ADDITIONAL_SCORES": [{"FULL_SCORE": 77}, {"INBOUND_FEAT_ID": 21, "INBOUND_FEAT_DESC": "15-MAR-1982", "CANDIDATE_FEAT_ID": 107, "CANDIDATE_FEAT_DESC": "17-OCT-1983", "SCORE_BUCKET": "NO_CHANCE"}]}, {"RECORD_ID": "DEF456", "NAME_FIRST": "Joann", "NAME_LAST": "Smith", "MOBILE_PHONE_NUMBER": "213-555-1212", "HOME_PHONE_NUMBER": "818-888-3939", "ADDR_FULL": "101 Fifth Ave, Los Angeles, CA 90018", "DATE_OF_BIRTH": "15-MAR-1982", "DATA_SOURCE": "PASSENGERS"}, {"RECORD_ID": "DEF456", "NAME_FIRST": "Joann Smith", "NAME_LAST": "Smith", "MOBILE_PHONE_NUMBER": "213-555-1212", "HOME_PHONE_NUMBER": "818-888-3939", "ADDR_FULL": "101 Fifth Ave, Los Angeles, CA 90018", "DATE_OF_BIRTH": "15-MAR-1982", "DATA_SOURCE": "PASSENGERS"}], "2025-10-17T23:21:04Z", "2025-10-17T23:21:04Z", {}, "CFF_SURNAME", "Joann Smith", 6, "Craig Smith", 7, 3, "POSSIBLY RELATED", "+ADDRESS-DOB", {"CONFIRMATIONS": [{"TOKEN": "ADDRESS", "FTYPE_CODE": "ADDRESS", "SCORE_BEHAVIOR": "FF", "SOURCE": "ADDRESS", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 108, "INBOUND_FEAT_DESC": "451 Dover Street, Los Angeles, CA 90018", "CANDIDATE_FEAT_ID": 108, "CANDIDATE_FEAT_DESC": "451 Dover Street, Los Angeles, CA 90018", "SCORE_BUCKET": "SAME"}], "DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 63, "ADDITIONAL_SCORES": [{"FULL_SCORE": 63}, {"INBOUND_FEAT_ID": 134, "INBOUND_FEAT_DESC": "24-NOV-1975", "CANDIDATE_FEAT_ID": 107, "CANDIDATE_FEAT_DESC": "17-OCT-1983", "SCORE_BUCKET": "NO_CHANCE"}]}, {"RECORD_ID": "ABC567", "NAME_FIRST": "Kim", "NAME_LAST": "Long", "MOBILE_PHONE_NUMBER": "818-246-8024", "HOME_PHONE_NUMBER": "818-135-7913", "ADDR_FULL": "451 Dover Street, Los Angeles, CA 90018", "DATE_OF_BIRTH": "24-NOV-1975", "DATA_SOURCE": "EMPLOYEES"}, {"RECORD_ID": "ABC567", "NAME_FIRST": "Kim Long", "NAME_LAST": "Long", "MOBILE_PHONE_NUMBER": "818-246-8024", "HOME_PHONE_NUMBER": "818-135-7913", "ADDR_FULL": "451 Dover Street, Los Angeles, CA 90018", "DATE_OF_BIRTH": "24-NOV-1975", "DATA_SOURCE": "EMPLOYEES"}], "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "SFF", "Kim Long", 7, "Kim Long", 0, 0, "", "", 0, 0, "EMPLOYEES", "ABC567", {"RECORD_ID": "ABC567", "NAME_FIRST": "Kim", "NAME_LAST": "Long", "MOBILE_PHONE_NUMBER": "818-246-8024", "HOME_PHONE_NUMBER": "818-135-7913", "ADDR_FULL": "451 Dover Street, Los Angeles, CA 90018", "DATE_OF_BIRTH": "24-NOV-1975", "DATA_SOURCE": "EMPLOYEES"}, {"RECORD_ID": "ABC567", "NAME_FIRST": "Kim Long", "NAME_LAST": "Long", "MOBILE_PHONE_NUMBER": "818-246-8024", "HOME_PHONE_NUMBER": "818-135-7913", "ADDR_FULL": "451 Dover Street, Los Angeles, CA 90018", "DATE_OF_BIRTH": "24-NOV-1975", "DATA_SOURCE": "EMPLOYEES"}], "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "", ""}]
```

7, "Kim Long", 6, 3, "POSSIBLY RELATED", "+ADDRESS-DOB", {"CONFIRMATIONS": [{"TOKEN": "ADDRESS", "FTYPE_CODE": "ADDRESS", "SCORE_BEHAVIOR": "FF", "SOURCE": "ADDRESS", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 108, "INBOUND_FEAT_DESC": "451 Dover Street, Los Angeles, CA"}], "CANDIDATE_FEAT_ID": 108, "CANDIDATE_FEAT_DESC": "451 Dover Street, Los Angeles, CA 90018", "SCORE_BUCKET": "SAME"}], "DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 63, "ADDITIONAL_SCORES": [{"FULL_SCORE": 63}], "INBOUND_FEAT_ID": 107, "INBOUND_FEAT_DESC": "17-OCT-1983", "CANDIDATE_FEAT_ID": 134, "CANDIDATE_FEAT_DESC": "24-NOV-1975", "SCORE_BUCKET": "NO_CHANCE"}]}, 0, 0, "EMPLOYEES", "PQR678", {"RECORD_ID": "PQR678", "NAME_FIRST": "Craig", "NAME_LAST": "Smith", "MOBILE_PHONE_NUMBER": "818-555-1212", "HOME_PHONE_NUMBER": "818-888-3939", "ADDR_FULL": "451 Dover Street, Los Angeles, CA 90018", "DATE_OF_BIRTH": "17-OCT-1983", "DATA_SOURCE": "EMPLOYEES"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "SFF", "Craig Smith"}]

8, "Katrina Osmond", 0, 0, "", "", 0, 0, "EMPLOYEES", "DEF890", {"RECORD_ID": "DEF890", "NAME_FIRST": "Katrina", "NAME_LAST": "Osmond", "MOBILE_PHONE_NUMBER": "818-444-2121", "HOME_PHONE_NUMBER": "818-111-2222", "ADDR_FULL": "707 Seventh Ave, Los Angeles, CA 90043", "DATE_OF_BIRTH": "27-JUN-1980", "DATA_SOURCE": "EMPLOYEES"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "", "", 5, 3, "POSSIBLY RELATED", "+PHONE-DOB", {"CONFIRMATIONS": [{"TOKEN": "PHONE", "FTYPE_CODE": "PHONE", "SCORE_BEHAVIOR": "F1", "SOURCE": "PHONE", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 83, "INBOUND_FEAT_DESC": "818-444-2121", "INBOUND_FEAT_USAGE_TYPE": "MOBILE", "CANDIDATE_FEAT_ID": 83, "CANDIDATE_FEAT_DESC": "818-444-2121", "CANDIDATE_FEAT_USAGE_TYPE": "MOBILE", "SCORE_BUCKET": "SAME"}], "DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 79, "ADDITIONAL_SCORES": [{"FULL_SCORE": 79}, {"INBOUND_FEAT_ID": 81, "INBOUND_FEAT_DESC": "22-AUG-1981", "CANDIDATE_FEAT_ID": 160, "CANDIDATE_FEAT_DESC": "27-JUN-1980", "SCORE_BUCKET": "NO_CHANCE"}]}, 0, 0, "EMPLOYEES", "MN0345", {"RECORD_ID": "MN0345", "NAME_FIRST": "Bill", "NAME_LAST": "Bandley", "MOBILE_PHONE_NUMBER": "818-444-2121", "HOME_PHONE_NUMBER": "818-123-4567", "ADDR_FULL": "101 Main Street, Los Angeles, CA 90011", "DATE_OF_BIRTH": "22-AUG-1981", "DATA_SOURCE": "EMPLOYEES"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "SF1", "Bill Bandley"}], "CONFIRMATIONS": [{"TOKEN": "ADDRESS", "FTYPE_CODE": "ADDRESS", "SCORE_BEHAVIOR": "FF", "SOURCE": "ADDRESS", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 161, "INBOUND_FEAT_DESC": "707 Seventh Ave, Los Angeles, CA 90043", "CANDIDATE_FEAT_ID": 161, "CANDIDATE_FEAT_DESC": "707"}]}]

Seventh Ave, Los Angeles, CA 90043", "SCORE_BUCKET": "SAME"}], "DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 68, "ADDITIONAL_SCORES": [{"FULL_SCORE": 68}, {"INBOUND_FEAT_ID": 236, "INBOUND_FEAT_DESC": "15-JAN-1979", "CANDIDATE_FEAT_ID": 160, "CANDIDATE_FEAT_DESC": "27-JUN-1980", "SCORE_BUCKET": "NO_CHANCE"}]}, {"RECORD_ID": "JKL456", "NAME_FIRST": "Kelly", "NAME_LAST": "Rogers", "MOBILE_PHONE_NUMBER": "818-333-7171", "HOME_PHONE_NUMBER": "818-789-6543", "ADDR_FULL": "707 Seventh Ave, Los Angeles, CA 90043", "DATE_OF_BIRTH": "15-JAN-1979", "DATA_SOURCE": "VIPS"}, {"RECORD_ID": "STU901", "NAME_FIRST": "Martha", "NAME_LAST": "Wayne", "MOBILE_PHONE_NUMBER": "818-891-9292", "HOME_PHONE_NUMBER": "818-987-1234", "ADDR_FULL": "888 Sepulveda Blvd, Los Angeles, CA 90034", "DATE_OF_BIRTH": "27-NOV-1973", "DATA_SOURCE": "VIPS"}, {"RECORD_ID": "STU901", "NAME_FIRST": "Martha", "NAME_LAST": "Wayne", "MOBILE_PHONE_NUMBER": "818-891-9292", "HOME_PHONE_NUMBER": "818-987-1234", "ADDR_FULL": "888 Sepulveda Blvd, Los Angeles, CA 90034", "DATE_OF_BIRTH": "27-NOV-1973", "DATA_SOURCE": "VIPS"}, {"RECORD_ID": "STU901", "NAME_FIRST": "Martha", "NAME_LAST": "Wayne", "MOBILE_PHONE_NUMBER": "818-891-9292", "HOME_PHONE_NUMBER": "818-987-1234", "ADDR_FULL": "888 Sepulveda Blvd, Los Angeles, CA 90034", "DATE_OF_BIRTH": "27-NOV-1973", "DATA_SOURCE": "VIPS"}, {"RECORD_ID": "GHI123", "NAME_FIRST": "Martha", "NAME_LAST": "Kent", "MOBILE_PHONE_NUMBER": "818-333-5757", "HOME_PHONE_NUMBER": "818-123-9876", "ADDR_FULL": "888 Sepulveda Blvd, Los Angeles, CA 90034", "DATE_OF_BIRTH": "17-AUG-1978", "DATA_SOURCE": "VIPS"}, {"RECORD_ID": "XYZ234", "NAME_FIRST": "Jane", "NAME_LAST": "Johnson", "MOBILE_PHONE_NUMBER": "818-333-7171", "HOME_PHONE_NUMBER": "818-123-9876", "ADDR_FULL": "400 River Street, Pasadena, CA 90034", "DATE_OF_BIRTH": "6-SEP-1975", "DATA_SOURCE": "VIPS"}]

10, "Jane Johnson", 4, 3, "POSSIBLY RELATED", "+PNAME+ADDRESS-DOB", {"CONFIRMATIONS": [{"TOKEN": "PNAME", "FTYPE_CODE": "NAME", "SCORE_BEHAVIOR": "NAME", "SOURCE": "/PART_NAME", "SCORE": 84, "ADDITIONAL_SCORES": [{"GNR_FN": 84, "GNR_SN": 68, "GNR_GN": 100, "GENERATION_MATCH": -1, "GNR_ON": -1}, {"INBOUND_FEAT_ID": 63, "INBOUND_FEAT_DESC": "Jane Donaldson", "CANDIDATE_FEAT_ID": 195, "CANDIDATE_FEAT_DESC": "Jane Johnson", "SCORE_BUCKET": "LIKELY"}, {"TOKEN": "ADDRESS", "FTYPE_CODE": "ADDRESS", "SCORE_BEHAVIOR": "FF", "SOURCE": "ADDRESS", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}], {"INBOUND_FEAT_ID": 65, "INBOUND_FEAT_DESC": "400 River Street, Pasadena, CA 90034", "CANDIDATE_FEAT_ID": 65, "CANDIDATE_FEAT_DESC": "400 River Street, Pasadena, CA 90034", "SCORE_BUCKET": "SAME"}], "DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 65, "ADDITIONAL_SCORES": [{"FULL_SCORE": 65}], {"INBOUND_FEAT_ID": 64, "INBOUND_FEAT_DESC": "23-MAY-1973", "CANDIDATE_FEAT_ID": 196, "CANDIDATE_FEAT_DESC": "6-SEP-1975", "SCORE_BUCKET": "NO_CHANCE"}]}], 0, 0, "PASSENGERS", "JKL012", {"RECORD_ID": "JKL012", "NAME_FIRST": "Jane", "NAME_LAST": "Donaldson", "MOBILE_PHONE_NUMBER": "818-555-1313", "HOME_PHONE_NUMBER": "818-222-3131", "ADDR_FULL": "400 River Street, Pasadena, CA 90034", "DATE_OF_BIRTH": "23-MAY-1973", "DATA_SOURCE": "PASSENGERS"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "SFF", "Jane Donaldson"}}, 10, "Jane Johnson", 11, 3, "POSSIBLY RELATED", "+PHONE-DOB", {"CONFIRMATIONS": [{"TOKEN": "PHONE", "FTYPE_CODE": "PHONE", "SCORE_BEHAVIOR": "FF", "SOURCE": "PHONE", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}], {"INBOUND_FEAT_ID": 197, "INBOUND_FEAT_DESC": "818-123-9876", "INBOUND_FEAT_USAGE_TYPE": "HOME", "CANDIDATE_FEAT_ID": 197, "CANDIDATE_FEAT_DESC": "818-123-9876", "SCORE_BUCKET": "SAME"}], "DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 67, "ADDITIONAL_SCORES": [{"FULL_SCORE": 67}], {"INBOUND_FEAT_ID": 222, "INBOUND_FEAT_DESC": "17-AUG-1978", "CANDIDATE_FEAT_ID": 196, "CANDIDATE_FEAT_DESC": "6-SEP-1975", "SCORE_BUCKET": "NO_CHANCE"}]}], 0, 0, "VIPS", "GHI123", {"RECORD_ID": "GHI123", "NAME_FIRST": "Martha", "NAME_LAST": "Kent", "MOBILE_PHONE_NUMBER": "818-333-5757", "HOME_PHONE_NUMBER": "818-123-9876", "ADDR_FULL": "888 Sepulveda Blvd, Los Angeles, CA 90034", "DATE_OF_BIRTH": "17-AUG-1978", "DATA_SOURCE": "VIPS"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "SFF", "Martha Kent"}}, 10, "Jane Johnson", 12, 3, "POSSIBLY RELATED", "+PHONE-DOB", {"CONFIRMATIONS": [{"TOKEN": "PHONE", "FTYPE_CODE": "PHONE", "SCORE_BEHAVIOR": "F1", "SOURCE": "PHONE", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}], {"INBOUND_FEAT_ID": 198, "INBOUND_FEAT_DESC": "818-333-7171", "INBOUND_FEAT_USAGE_TYPE": "MOBILE", "CANDIDATE_FEAT_ID": 198, "CANDIDATE_FEAT_DESC": "818-333-7171"}]}]

7171", "CANDIDATE_FEAT_USAGE_TYPE": "MOBILE", "SCORE_BUCKET": "SAME"}], "DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 72, "ADDITIONAL_SCORES": [{"FULL_SCORE": 72}, {"INBOUND_FEAT_ID": 236, "INBOUND_FEAT_DESC": "15-JAN-1979"}, {"CANDIDATE_FEAT_ID": 196, "CANDIDATE_FEAT_DESC": "6-SEP-1975"}, {"SCORE_BUCKET": "NO_CHANCE"}], 0, 0, "VIPS", "JKL456", {"RECORD_ID": "JKL456", "NAME_FIRST": "Kelly", "NAME_LAST": "Rogers", "MOBILE_PHONE_NUMBER": "818-333-7171", "HOME_PHONE_NUMBER": "818-789-6543", "ADDR_FULL": "707 Seventh Ave, Los Angeles, CA", "DATE_OF_BIRTH": "15-JAN-1979", "DATA_SOURCE": "VIPS"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "SF1", "Kelly Rogers", 11, "Martha Kent", 0, 0, "", "", 0, 0, "VIPS", "GHI123", {"RECORD_ID": "GHI123", "NAME_FIRST": "Martha", "NAME_LAST": "Kent", "MOBILE_PHONE_NUMBER": "818-333-5757", "HOME_PHONE_NUMBER": "818-123-9876", "ADDR_FULL": "888 Sepulveda Blvd, Los Angeles, CA", "DATE_OF_BIRTH": "17-AUG-1978", "DATA_SOURCE": "VIPS"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "", 11, "Martha Kent", 9, 3, "POSSIBLY RELATED", "+PNAME+ADDRESS-DOB", {"CONFIRMATIONS": [{"TOKEN": "PNAME", "FTYPE_CODE": "NAME", "SCORE_BEHAVIOR": "NAME", "SOURCE": "/PART_NAME", "SCORE": 68, "ADDITIONAL_SCORES": [{"GNR_FN": 68, "GNR_SN": 34, "GNR_GN": 100, "GENERATION_MATCH": -1, "GNR_ON": -1}, {"INBOUND_FEAT_ID": 176, "INBOUND_FEAT_DESC": "Martha Wayne"}, {"CANDIDATE_FEAT_ID": 221, "CANDIDATE_FEAT_DESC": "Martha Kent", "SCORE_BUCKET": "UNLIKELY"}], {"ADDRESS": "888 Sepulveda Blvd, Los Angeles, CA", "FTYPE_CODE": "ADDRESS", "SCORE_BEHAVIOR": "FF", "SOURCE": "ADDRESS", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}], {"INBOUND_FEAT_ID": 178, "INBOUND_FEAT_DESC": "888 Sepulveda Blvd, Los Angeles, CA", "CANDIDATE_FEAT_ID": 178, "CANDIDATE_FEAT_DESC": "888 Sepulveda Blvd, Los Angeles, CA", "SCORE_BUCKET": "SAME"}], "DENIALS": [{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "DOB", "SCORE": 67, "ADDITIONAL_SCORES": [{"FULL_SCORE": 67}, {"INBOUND_FEAT_ID": 177, "INBOUND_FEAT_DESC": "27-NOV-1973"}, {"CANDIDATE_FEAT_ID": 222, "CANDIDATE_FEAT_DESC": "17-AUG-1978"}, {"SCORE_BUCKET": "NO_CHANCE"}], 0, 0, "VIPS", "STU901", {"RECORD_ID": "STU901", "NAME_FIRST": "Martha", "NAME_LAST": "Wayne", "MOBILE_PHONE_NUMBER": "818-891-9292", "HOME_PHONE_NUMBER": "818-987-1234", "ADDR_FULL": "888 Sepulveda Blvd, Los Angeles, CA", "DATE_OF_BIRTH": "27-NOV-1973", "DATA_SOURCE": "VIPS"}, "2025-10-17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "SFF", "Martha Wayne", 11, "Martha Kent", 10, 3, "POSSIBLY RELATED", "+PHONE-DOB", {"CONFIRMATIONS": [{"TOKEN": "PHONE", "FTYPE_CODE": "PHONE", "SCORE_BEHAVIOR": "FF", "SOURCE": "PHONE", "SCORE": 100, "ADDITIONAL_SCORES": [{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 197, "INBOUND_FEAT_DESC": "818-123-9876"}, {"INBOUND_FEAT_USAGE_TYPE": "HOME", "CANDIDATE_FEAT_ID": 197, "CANDIDATE_FEAT_DESC": "818-123-9876"}]}]}]}]

AT_DESC": "818-123-
9876", "CANDIDATE_FEAT_USAGE_TYPE": "HOME", "SCORE_BUCKET": "SAME"}], "DENIALS":
":
[{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "
DOB", "SCORE": 67, "ADDITIONAL_SCORES":
{"FULL_SCORE": 67}, {"INBOUND_FEAT_ID": 196, "INBOUND_FEAT_DESC": "6-SEP-
1975", "CANDIDATE_FEAT_ID": 222, "CANDIDATE_FEAT_DESC": "17-AUG-
1978", "SCORE_BUCKET": "NO_CHANCE"}], 0, 0, "VIPS", "XYZ234", "
{"RECORD_ID": "XYZ234", "NAME_FIRST": "Jane", "NAME_LAST": "Johnson", "MOBILE
_PHONE_NUMBER": "818-333-7171", "HOME_PHONE_NUMBER": "818-123-
9876", "ADDR_FULL": "400 River Street, Pasadena, CA 90034", "DATE_OF_BIRTH": "6-
SEP-1975", "DATA_SOURCE": "VIPS"}, "2025-10-17T23:21:14Z", "2025-10-
17T23:21:14Z", {}, "SFF", "Jane Johnson"
12, "Kelly Rogers", 0, 0, "", "", 0, 0, "VIPS", "JKL456", "
{"RECORD_ID": "JKL456", "NAME_FIRST": "Kelly", "NAME_LAST": "Rogers", "MOBILE
_PHONE_NUMBER": "818-333-7171", "HOME_PHONE_NUMBER": "818-789-
6543", "ADDR_FULL": "707 Seventh Ave, Los Angeles, CA
90043", "DATE_OF_BIRTH": "15-JAN-1979", "DATA_SOURCE": "VIPS"}, "2025-10-
17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "", "
12, "Kelly Rogers", 8, 3, "POSSIBLY RELATED", "+ADDRESS-DOB", {"CONFIRMATIONS":
[{"TOKEN": "ADDRESS", "FTYPE_CODE": "ADDRESS", "SCORE_BEHAVIOR": "FF", "SOURCE": "
ADDRESS", "SCORE": 100, "ADDITIONAL_SCORES":
{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 161, "INBOUND_FEAT_DESC": "707 Seventh
Ave, Los Angeles, CA 90043", "CANDIDATE_FEAT_ID": 161, "CANDIDATE_FEAT_DESC": "707
Seventh Ave, Los Angeles, CA 90043", "SCORE_BUCKET": "SAME"}], "DENIALS":
[{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "
DOB", "SCORE": 68, "ADDITIONAL_SCORES":
{"FULL_SCORE": 68}, {"INBOUND_FEAT_ID": 160, "INBOUND_FEAT_DESC": "27-JUN-
1980", "CANDIDATE_FEAT_ID": 236, "CANDIDATE_FEAT_DESC": "15-JAN-
1979", "SCORE_BUCKET": "NO_CHANCE"}], 0, 0, "EMPLOYEES", "DEF890", "
{"RECORD_ID": "DEF890", "NAME_FIRST": "Katrina", "NAME_LAST": "Osmond", "MOBI
LE_PHONE_NUMBER": "818-444-2121", "HOME_PHONE_NUMBER": "818-111-
2222", "ADDR_FULL": "707 Seventh Ave, Los Angeles, CA
90043", "DATE_OF_BIRTH": "27-JUN-1980", "DATA_SOURCE": "EMPLOYEES"}, "2025-10-
17T23:21:14Z", "2025-10-17T23:21:14Z", {}, "SFF", "Katrina Osmond"
12, "Kelly Rogers", 10, 3, "POSSIBLY RELATED", "+PHONE-DOB", {"CONFIRMATIONS":
[{"TOKEN": "PHONE", "FTYPE_CODE": "PHONE", "SCORE_BEHAVIOR": "F1", "SOURCE": "
PHONE", "SCORE": 100, "ADDITIONAL_SCORES":
{"FULL_SCORE": 100}, {"INBOUND_FEAT_ID": 198, "INBOUND_FEAT_DESC": "818-333-
7171", "INBOUND_FEAT_USAGE_TYPE": "MOBILE", "CANDIDATE_FEAT_ID": 198, "CANDIDATE
_FEAT_DESC": "818-333-
7171", "CANDIDATE_FEAT_USAGE_TYPE": "MOBILE", "SCORE_BUCKET": "SAME"}], "DENIAL
S":
[{"TOKEN": "DOB", "FTYPE_CODE": "DOB", "SCORE_BEHAVIOR": "FMES", "SOURCE": "
DOB", "SCORE": 72, "ADDITIONAL_SCORES":
{"FULL_SCORE": 72}, {"INBOUND_FEAT_ID": 196, "INBOUND_FEAT_DESC": "6-SEP-
1975", "CANDIDATE_FEAT_ID": 222, "CANDIDATE_FEAT_DESC": "17-AUG-
1978", "SCORE_BUCKET": "NO_CHANCE"}]

```
1975 "", ""CANDIDATE_FEAT_ID"":236, ""CANDIDATE_FEAT_DESC"":""15-JAN-  
1979 "", ""SCORE_BUCKET"":""NO_CHANCE""}]}}},0,0,"VIPS","XYZ234",  
{""RECORD_ID"":""XYZ234""}, ""NAME_FIRST"":""Jane""", ""NAME_LAST"":""Johnson""", ""MOBILE  
_PHONE_NUMBER"":""818-333-7171""", ""HOME_PHONE_NUMBER"":""818-123-  
9876 "", ""ADDR_FULL"":""400 River Street, Pasadena, CA 90034""", ""DATE_OF_BIRTH"":""6-  
SEP-1975 "", ""DATA_SOURCE"":""VIPS""}", "2025-10-17T23:21:14Z", "2025-10-  
17T23:21:14Z", "{}", "SF1", "Jane Johnson"  
100002, "Joe Schmoe", 0, 0, "", "", 0, 0, "TEST", "ABC123", "  
{""DATA_SOURCE"":""TEST""}, ""RECORD_ID"":""ABC123""}, ""NAME_FULL"":""Joe  
Schmoe""", ""PHONE_NUMBER"":""702-555-  
1212""", ""EMAIL_ADDRESS"":""joeschmoe@nowhere.com""}", "2025-10-17T23:21:14Z", "2025-  
10-17T23:21:14Z", "{}", "", ""
```

Remarks

Used in conjunction with [FetchNext\(IntPtr\)](#) and [CloseExportReport\(IntPtr\)](#). The first [FetchNext\(IntPtr\)](#) call after calling this method returns the CSV header. Subsequent [FetchNext\(IntPtr\)](#) calls return exported entity data in CSV format.

WARNING: This method should only be used on systems containing less than a few million records. For larger systems, see

https://www.senzing.com/docs/tutorials/advanced_replication/.

The optionally specified bitwise-OR'd [SzFlag](#) values not only control how the operation is performed but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzExportFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzException](#)

If a failure occurs.

See Also

[SzExportDefaultFlags](#), [SzExportFlags](#), [FetchNext\(IntPtr\)](#), [CloseExportReport\(IntPtr\)](#), [ExportJsonEntityReport\(SzFlag?\)](#)

ExportJsonEntityReport(SzFlag?)

Initiates an export report of entity data in JSON Lines format.

```
[SzConfigRetryable]
IntPtr ExportJsonEntityReport(SzFlag? flags =
SzFlag.SzExportIncludeMultiRecordEntities |
SzFlag.SzExportIncludeSingleRecordEntities |
SzFlag.SzEntityIncludePossiblySameRelations |
SzFlag.SzEntityIncludePossiblyRelatedRelations |
SzFlag.SzEntityIncludeNameOnlyRelations | SzFlag.SzEntityIncludeDisclosedRelations |
SzFlag.SzEntityIncludeRepresentativeFeatures | SzFlag.SzEntityIncludeEntityName |
SzFlag.SzEntityIncludeRecordSummary | SzFlag.SzEntityIncludeRecordData |
SzFlag.SzEntityIncludeRecordMatchingInfo | SzFlag.SzEntityIncludeRelatedEntityName |
SzFlag.SzEntityIncludeRelatedMatchingInfo |
SzFlag.SzEntityIncludeRelatedRecordSummary)
```

Parameters

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzExportFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter defaults it value to [SzExportDefaultFlags](#) for the default recommended flags. Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[IntPtr](#)

The export handle to use for retrieving the export data.

Examples

Usage:

```
// How to export entity data in JSON format
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // export the JSON data
    IntPtr exportHandle = engine.ExportJsonEntityReport(SzExportDefaultFlags);
```

```

// read the data
try
{
    // fetch the first JSON record
    string jsonData = engine.FetchNext(exportHandle);

    while (jsonData != null)
    {
        // parse the JSON data
        JsonObject? jsonObject = JsonNode.Parse(jsonData)?.AsObject();

        // do something with the parsed data (varies by application)
        ProcessJsonRecord(jsonObject);

        // fetch the next JSON record
        jsonData = engine.FetchNext(exportHandle);
    }

}

finally
{
    // close the export handle
    engine.CloseExportReport(exportHandle);
}

}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to perform JSON export.", e);
}

```

Example Complete Export Report:

```

{"RESOLVED_ENTITY": {"ENTITY_ID": 1, "ENTITY_NAME": "Joseph Schmidt", "FEATURES": [{"ADDRESS": [{"FEAT_DESC": "101 Main Street, Los Angeles, CA 90011", "LIB_FEAT_ID": 3, "FEAT_DESC_VALUES": [{"FEAT_DESC": "101 Main Street, Los Angeles, CA 90011", "LIB_FEAT_ID": 3}]}], "DOB": [{"FEAT_DESC": "12-JAN-1981", "LIB_FEAT_ID": 2, "FEAT_DESC_VALUES": [{"FEAT_DESC": "12-JAN-1981", "LIB_FEAT_ID": 2}]}], "NAME": [{"FEAT_DESC": "Joseph Schmidt", "LIB_FEAT_ID": 1, "FEAT_DESC_VALUES": [{"FEAT_DESC": "Joseph Schmidt", "LIB_FEAT_ID": 1}]}], "PHONE": [{"FEAT_DESC": "818-777-2424", "LIB_FEAT_ID": 4, "USAGE_TYPE": "HOME", "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-777-2424", "LIB_FEAT_ID": 4}], "FEAT_DESC": "213-555-"}]}

```

1212", "LIB_FEAT_ID":5, "USAGE_TYPE":"MOBILE", "FEAT_DESC_VALUES": [{"FEAT_DESC":"213-555-1212", "LIB_FEAT_ID":5}]]}], "RECORD_SUMMARY": [{"DATA_SOURCE":"PASSENGERS", "RECORD_COUNT":1}], "RECORDS": [{"DATA_SOURCE":"PASSENGERS", "RECORD_ID":"ABC123", "INTERNAL_ID":1, "MATCH_KEY": "", "MATCH_LEVEL_CODE": "", "ERRULE_CODE":""}], "RELATED_ENTITIES": [{"ENTITY_ID":2, "MATCH_LEVEL_CODE":"POSSIBLY RELATED", "MATCH_KEY": "+PHONE-DOB", "ERRULE_CODE":"SF1", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Joann Smith", "RECORD_SUMMARY": [{"DATA_SOURCE": "PASSENGERS", "RECORD_COUNT": 1}]}, {"ENTITY_ID":5, "MATCH_LEVEL_CODE":"POSSIBLY RELATED", "MATCH_KEY": "+ADDRESS-DOB", "ERRULE_CODE": "SFF", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Bill Bandley", "RECORD_SUMMARY": [{"DATA_SOURCE": "EMPLOYEES", "RECORD_COUNT": 1}]}], {"RESOLVED_ENTITY": {"ENTITY_ID":2, "ENTITY_NAME": "Joann Smith", "FEATURES": {"ADDRESS": [{"FEAT_DESC": "101 Fifth Ave, Los Angeles, CA 90018", "LIB_FEAT_ID":22, "FEAT_DESC_VALUES": [{"FEAT_DESC": "101 Fifth Ave, Los Angeles, CA 90018", "LIB_FEAT_ID":22}]}], "DOB": [{"FEAT_DESC": "15-MAR-1982", "LIB_FEAT_ID":21, "FEAT_DESC_VALUES": [{"FEAT_DESC": "15-MAR-1982", "LIB_FEAT_ID":21}]}], "NAME": [{"FEAT_DESC": "Joann Smith", "LIB_FEAT_ID":20, "FEAT_DESC_VALUES": [{"FEAT_DESC": "Joann Smith", "LIB_FEAT_ID":20}]}], "PHONE": [{"FEAT_DESC": "818-888-3939", "LIB_FEAT_ID":23, "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-888-3939", "LIB_FEAT_ID":23}]}]}, {"FEAT_DESC": "213-555-1212", "LIB_FEAT_ID":5, "FEAT_DESC_VALUES": [{"FEAT_DESC": "213-555-1212", "LIB_FEAT_ID":5}]]}], "RECORD_SUMMARY": [{"DATA_SOURCE": "PASSENGERS", "RECORD_COUNT": 1}], "RECORDS": [{"DATA_SOURCE": "PASSENGERS", "RECORD_ID": "DEF456", "INTERNAL_ID": 2, "MATCH_KEY": "", "MATCH_LEVEL_CODE": "", "ERRULE_CODE": ""}], "RELATED_ENTITIES": [{"ENTITY_ID":1, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+PHONE-DOB", "ERRULE_CODE": "SF1", "IS_DISCLOSED": 0, "IS_AMBIGUOUS": 0, "ENTITY_NAME": "Joseph Schmidt", "RECORD_SUMMARY": [{"DATA_SOURCE": "PASSENGERS", "RECORD_COUNT": 1}]}, {"ENTITY_ID":3, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+ADDRESS-DOB", "ERRULE_CODE": "SFF", "IS_DISCLOSED": 0, "IS_AMBIGUOUS": 0, "ENTITY_NAME": "John Parker", "RECORD_SUMMARY": [{"DATA_SOURCE": "PASSENGERS", "RECORD_COUNT": 1}]}, {"ENTITY_ID":6, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+SURNAME+PHONE-DOB", "ERRULE_CODE": "CFF_SURNAME", "IS_DISCLOSED": 0, "IS_AMBIGUOUS": 0, "ENTITY_NAME": "Craig Smith", "RECORD_SUMMARY": [{"DATA_SOURCE": "EMPLOYEES", "RECORD_COUNT": 1}]}], {"RESOLVED_ENTITY": {"ENTITY_ID":3, "ENTITY_NAME": "John Parker", "FEATURES": {"ADDRESS": [{"FEAT_DESC": "101 Fifth Ave, Los Angeles, CA 90018", "LIB_FEAT_ID":22, "FEAT_DESC_VALUES": [{"FEAT_DESC": "101 Fifth Ave, Los Angeles, CA 90018", "LIB_FEAT_ID":22}]}], "DOB": [{"FEAT_DESC": "17-DEC-1977", "LIB_FEAT_ID":48, "FEAT_DESC_VALUES": [{"FEAT_DESC": "17-DEC-1977", "LIB_FEAT_ID":48}]}], "NAME": [{"FEAT_DESC": "John Parker", "LIB_FEAT_ID":47, "FEAT_DESC_VALUES": [{"FEAT_DESC": "John Parker", "LIB_FEAT_ID":47}]}], "PHONE": [{"FEAT_DESC": "818-999-2121", "LIB_FEAT_ID":49, "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-999-2121", "LIB_FEAT_ID":49}]}]}, {"FEAT_DESC": "818-555-1313", "LIB_FEAT_ID":50, "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-555-1313", "LIB_FEAT_ID":50}]}]}

555-1313", "LIB_FEAT_ID":50}]]}], "RECORD_SUMMARY": [{"DATA_SOURCE": "PASSENGERS", "RECORD_COUNT":1}], "RECORDS": [{"DATA_SOURCE": "PASSENGERS", "RECORD_ID": "GHI789", "INTERNAL_ID":3, "MATCH_KEY": "", "MATCH_LEVEL_CODE": "", "ERRULE_CODE": ""}], "RELATED_ENTITIES": [{"ENTITY_ID":2, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+ADDRESS-DOB", "ERRULE_CODE": "SFF", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Joann Smith", "RECORD_SUMMARY": [{"DATA_SOURCE": "PASSENGERS", "RECORD_COUNT":1}]}, {"ENTITY_ID":4, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+PHONE-DOB", "ERRULE_CODE": "SF1", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Jane Donaldson", "RECORD_SUMMARY": [{"DATA_SOURCE": "PASSENGERS", "RECORD_COUNT":1}]}]}, {"RESOLVED_ENTITY": {"ENTITY_ID":4, "ENTITY_NAME": "Jane Donaldson", "FEATURES": {"ADDRESS": [{"FEAT_DESC": "400 River Street, Pasadena, CA 90034", "LIB_FEAT_ID":65, "FEAT_DESC_VALUES": [{"FEAT_DESC": "400 River Street, Pasadena, CA 90034", "LIB_FEAT_ID":65}]}, {"DOB": [{"FEAT_DESC": "23-MAY-1973", "LIB_FEAT_ID":64, "FEAT_DESC_VALUES": [{"FEAT_DESC": "23-MAY-1973", "LIB_FEAT_ID":64}]}, {"NAME": [{"FEAT_DESC": "Jane Donaldson", "LIB_FEAT_ID":63, "FEAT_DESC_VALUES": [{"FEAT_DESC": "Jane Donaldson", "LIB_FEAT_ID":63}]}, {"PHONE": [{"FEAT_DESC": "818-222-3131", "LIB_FEAT_ID":66, "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-222-3131", "LIB_FEAT_ID":66}]}, {"{"FEAT_DESC": "818-555-1313", "LIB_FEAT_ID":50, "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-555-1313", "LIB_FEAT_ID":50}]}], "RECORD_SUMMARY": [{"DATA_SOURCE": "PASSENGERS", "RECORD_COUNT":1}], "RECORDS": [{"DATA_SOURCE": "PASSENGERS", "RECORD_ID": "JKL012", "INTERNAL_ID":4, "MATCH_KEY": "", "MATCH_LEVEL_CODE": "", "ERRULE_CODE": ""}], "RELATED_ENTITIES": [{"ENTITY_ID":3, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+PHONE-DOB", "ERRULE_CODE": "SF1", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "John Parker", "RECORD_SUMMARY": [{"DATA_SOURCE": "PASSENGERS", "RECORD_COUNT":1}]}, {"ENTITY_ID":10, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+PNAME+ADDRESS-DOB", "ERRULE_CODE": "SFF", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Jane Johnson", "RECORD_SUMMARY": [{"DATA_SOURCE": "VIPS", "RECORD_COUNT":1}]}]}, {"RESOLVED_ENTITY": {"ENTITY_ID":5, "ENTITY_NAME": "Bill Bandley", "FEATURES": {"ADDRESS": [{"FEAT_DESC": "101 Main Street, Los Angeles, CA 90011", "LIB_FEAT_ID":3, "FEAT_DESC_VALUES": [{"FEAT_DESC": "101 Main Street, Los Angeles, CA 90011", "LIB_FEAT_ID":3}]}, {"DOB": [{"FEAT_DESC": "22-AUG-1981", "LIB_FEAT_ID":81, "FEAT_DESC_VALUES": [{"FEAT_DESC": "22-AUG-1981", "LIB_FEAT_ID":81}]}, {"NAME": [{"FEAT_DESC": "Bill Bandley", "LIB_FEAT_ID":80, "FEAT_DESC_VALUES": [{"FEAT_DESC": "Bill Bandley", "LIB_FEAT_ID":80}]}, {"PHONE": [{"FEAT_DESC": "818-123-4567", "LIB_FEAT_ID":82, "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-123-4567", "LIB_FEAT_ID":82}]}, {"{"FEAT_DESC": "818-444-2121", "LIB_FEAT_ID":83, "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-444-2121", "LIB_FEAT_ID":83}]}], "RECORD_SUMMARY": [{"DATA_SOURCE": "EMPLOYEES", "RECORD_COUNT":1}], "RECORDS": [{"DATA_SOURCE": "EMPLOYEES", "RECORD_ID": "MN0345", "INTERNAL_ID":5, "MATCH_KEY": "", "MATCH_LEVEL_CODE": "", "ERRULE_CODE": ""}], "RELATED_ENTITIES": []}]}]}]

```
[{"ENTITY_ID":1,"MATCH_LEVEL_CODE":"POSSIBLY RELATED","MATCH_KEY": "+ADDRESS-DOB","ERRULE_CODE":"SFF","IS_DISCLOSED":0,"IS_AMBIGUOUS":0,"ENTITY_NAME":"Joseph Schmidt","RECORD_SUMMARY":[{"DATA_SOURCE":"PASSENGERS","RECORD_COUNT":1}]}], {"ENTITY_ID":8,"MATCH_LEVEL_CODE":"POSSIBLY RELATED","MATCH_KEY": "+PHONE-DOB","ERRULE_CODE":"SF1","IS_DISCLOSED":0,"IS_AMBIGUOUS":0,"ENTITY_NAME":"Katrina Osmond","RECORD_SUMMARY":[{"DATA_SOURCE":"EMPLOYEES","RECORD_COUNT":1}]}]} {"RESOLVED_ENTITY":{"ENTITY_ID":6,"ENTITY_NAME":"Craig Smith","FEATURES":{"ADDRESS": [{"FEAT_DESC":"451 Dover Street, Los Angeles, CA 90018","LIB_FEAT_ID":108,"FEAT_DESC_VALUES": [{"FEAT_DESC":"451 Dover Street, Los Angeles, CA 90018","LIB_FEAT_ID":108}]}],"DOB": [{"FEAT_DESC": "17-OCT-1983","LIB_FEAT_ID":107,"FEAT_DESC_VALUES": [{"FEAT_DESC": "17-OCT-1983","LIB_FEAT_ID":107}]}],"NAME": [{"FEAT_DESC": "Craig Smith","LIB_FEAT_ID":106,"FEAT_DESC_VALUES": [{"FEAT_DESC": "Craig Smith","LIB_FEAT_ID":106}]}],"PHONE": [{"FEAT_DESC": "818-888-3939","LIB_FEAT_ID":23,"FEAT_DESC_VALUES": [{"FEAT_DESC": "818-888-3939","LIB_FEAT_ID":23}]}], {"FEAT_DESC": "818-555-1212","LIB_FEAT_ID":109,"FEAT_DESC_VALUES": [{"FEAT_DESC": "818-555-1212","LIB_FEAT_ID":109}]}]}],"RECORD_SUMMARY": [{"DATA_SOURCE": "EMPLOYEES","RECORD_COUNT":1}]}], "RECORDS": [{"DATA_SOURCE": "EMPLOYEES","RECORD_ID": "PQR678","INTERNAL_ID":6,"MATCH_KEY": "", "MATCH_LEVEL_CODE": "", "ERRULE_CODE": ""}], "RELATED_ENTITIES": [{"ENTITY_ID":2,"MATCH_LEVEL_CODE":"POSSIBLY RELATED","MATCH_KEY": "+SURNAME+PHONE-DOB","ERRULE_CODE":"CFF_SURNAME","IS_DISCLOSED":0,"IS_AMBIGUOUS":0,"ENTITY_NAME":"Joann Smith","RECORD_SUMMARY":[{"DATA_SOURCE": "PASSENGERS","RECORD_COUNT":1}]}], {"ENTITY_ID":7,"MATCH_LEVEL_CODE":"POSSIBLY RELATED","MATCH_KEY": "+ADDRESS-DOB","ERRULE_CODE":"SFF","IS_DISCLOSED":0,"IS_AMBIGUOUS":0,"ENTITY_NAME":"Kim Long","RECORD_SUMMARY":[{"DATA_SOURCE": "EMPLOYEES","RECORD_COUNT":1}]}]} {"RESOLVED_ENTITY":{"ENTITY_ID":7,"ENTITY_NAME":"Kim Long","FEATURES":{"ADDRESS": [{"FEAT_DESC": "451 Dover Street, Los Angeles, CA 90018","LIB_FEAT_ID":108,"FEAT_DESC_VALUES": [{"FEAT_DESC": "451 Dover Street, Los Angeles, CA 90018","LIB_FEAT_ID":108}]}],"DOB": [{"FEAT_DESC": "24-NOV-1975","LIB_FEAT_ID":134,"FEAT_DESC_VALUES": [{"FEAT_DESC": "24-NOV-1975","LIB_FEAT_ID":134}]}],"NAME": [{"FEAT_DESC": "Kim Long","LIB_FEAT_ID":133,"FEAT_DESC_VALUES": [{"FEAT_DESC": "Kim Long","LIB_FEAT_ID":133}]}],"PHONE": [{"FEAT_DESC": "818-135-7913","LIB_FEAT_ID":135,"FEAT_DESC_VALUES": [{"FEAT_DESC": "818-135-7913","LIB_FEAT_ID":135}]}], {"FEAT_DESC": "818-246-8024","LIB_FEAT_ID":136,"FEAT_DESC_VALUES": [{"FEAT_DESC": "818-246-8024","LIB_FEAT_ID":136}]}]}],"RECORD_SUMMARY": [{"DATA_SOURCE": "EMPLOYEES","RECORD_COUNT":1}]}], "RECORDS": [{"DATA_SOURCE": "EMPLOYEES","RECORD_ID": "ABC567","INTERNAL_ID":7,"MATCH_KEY": "", "MATCH_LEVEL_CODE": "", "ERRULE_CODE": ""}], "RELATED_ENTITIES": [{"ENTITY_ID":6,"MATCH_LEVEL_CODE":"POSSIBLY RELATED","MATCH_KEY": "+ADDRESS-DOB","ERRULE_CODE":"SFF","IS_DISCLOSED":0,"IS_AMBIGUOUS":0,"ENTITY_NAME":"Craig Smith","RECORD_SUMMARY":[{"DATA_SOURCE": "EMPLOYEES","RECORD_COUNT":1}]}]} {"RESOLVED_ENTITY":{"ENTITY_ID":8,"ENTITY_NAME":"Katrina Osmond","FEATURES":
```

```
{"ADDRESS":[{"FEAT_DESC":"707 Seventh Ave, Los Angeles, CA 90043","LIB_FEAT_ID":161,"FEAT_DESC_VALUES":[{"FEAT_DESC":"707 Seventh Ave, Los Angeles, CA 90043","LIB_FEAT_ID":161}]},{ "DOB": [{"FEAT_DESC":"27-JUN-1980","LIB_FEAT_ID":160,"FEAT_DESC_VALUES":[{"FEAT_DESC":"27-JUN-1980","LIB_FEAT_ID":160}]},{ "NAME": [{"FEAT_DESC":"Katrina Osmond","LIB_FEAT_ID":159,"FEAT_DESC_VALUES":[{"FEAT_DESC":"Katrina Osmond","LIB_FEAT_ID":159}]},{ "PHONE": [{"FEAT_DESC":"818-111-2222","LIB_FEAT_ID":162,"USAGE_TYPE": "HOME", "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-111-2222", "LIB_FEAT_ID": 162}], {"FEAT_DESC": "818-444-2121", "LIB_FEAT_ID": 83, "USAGE_TYPE": "MOBILE", "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-444-2121", "LIB_FEAT_ID": 83}]}]}, "RECORD_SUMMARY": [{"DATA_SOURCE": "EMPLOYEES", "RECORD_COUNT": 1}], "RECORDS": [{"DATA_SOURCE": "EMPLOYEES", "RECORD_ID": "DEF890", "INTERNAL_ID": 8, "MATCH_KEY": "", "MATCH_LEVEL_CODE": "", "ERRULE_CODE": ""}], "RELATED_ENTITIES": [{"ENTITY_ID": 5, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+PHONE-DOB", "ERRULE_CODE": "SF1", "IS_DISCLOSED": 0, "IS_AMBIGUOUS": 0, "ENTITY_NAME": "Bill Bandley", "RECORD_SUMMARY": [{"DATA_SOURCE": "EMPLOYEES", "RECORD_COUNT": 1}], {"ENTITY_ID": 12, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+ADDRESS-DOB", "ERRULE_CODE": "SFF", "IS_DISCLOSED": 0, "IS_AMBIGUOUS": 0, "ENTITY_NAME": "Kelly Rogers", "RECORD_SUMMARY": [{"DATA_SOURCE": "VIPS", "RECORD_COUNT": 1}]}], "RESOLVED_ENTITY": {"ENTITY_ID": 9, "ENTITY_NAME": "Martha Wayne", "FEATURES": {"ADDRESS": [{"FEAT_DESC": "888 Sepulveda Blvd, Los Angeles, CA 90034", "LIB_FEAT_ID": 178, "FEAT_DESC_VALUES": [{"FEAT_DESC": "888 Sepulveda Blvd, Los Angeles, CA 90034", "LIB_FEAT_ID": 178}]}, {"DOB": [{"FEAT_DESC": "27-NOV-1973", "LIB_FEAT_ID": 177, "FEAT_DESC_VALUES": [{"FEAT_DESC": "27-NOV-1973", "LIB_FEAT_ID": 177}]}, {"NAME": [{"FEAT_DESC": "Martha Wayne", "LIB_FEAT_ID": 176, "FEAT_DESC_VALUES": [{"FEAT_DESC": "Martha Wayne", "LIB_FEAT_ID": 176}]}, {"PHONE": [{"FEAT_DESC": "818-987-1234", "LIB_FEAT_ID": 179, "USAGE_TYPE": "HOME", "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-987-1234", "LIB_FEAT_ID": 179}], {"FEAT_DESC": "818-891-9292", "LIB_FEAT_ID": 180, "USAGE_TYPE": "MOBILE", "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-891-9292", "LIB_FEAT_ID": 180}}]}], "RECORD_SUMMARY": [{"DATA_SOURCE": "VIPS", "RECORD_COUNT": 1}], "RECORDS": [{"DATA_SOURCE": "VIPS", "RECORD_ID": "STU901", "INTERNAL_ID": 9, "MATCH_KEY": "", "MATCH_LEVEL_CODE": "", "ERRULE_CODE": ""}], "RELATED_ENTITIES": [{"ENTITY_ID": 11, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+PNAME+ADDRESS-DOB", "ERRULE_CODE": "SFF", "IS_DISCLOSED": 0, "IS_AMBIGUOUS": 0, "ENTITY_NAME": "Martha Kent", "RECORD_SUMMARY": [{"DATA_SOURCE": "VIPS", "RECORD_COUNT": 1}]}], "RESOLVED_ENTITY": {"ENTITY_ID": 10, "ENTITY_NAME": "Jane Johnson", "FEATURES": {"ADDRESS": [{"FEAT_DESC": "400 River Street, Pasadena, CA 90034", "LIB_FEAT_ID": 65, "FEAT_DESC_VALUES": [{"FEAT_DESC": "400 River Street, Pasadena, CA 90034", "LIB_FEAT_ID": 65}]}, {"DOB": [{"FEAT_DESC": "6-SEP-1975", "LIB_FEAT_ID": 196, "FEAT_DESC_VALUES": [{"FEAT_DESC": "6-SEP-1975", "LIB_FEAT_ID": 196}]}, {"NAME": [{"FEAT_DESC": "Jane Johnson", "LIB_FEAT_ID": 195, "FEAT_DESC_VALUES": [{"FEAT_DESC": "Jane Johnson", "LIB_FEAT_ID": 195}]}, {"PHONE": [{"FEAT_DESC": "818-123-"}]}]}]}
```

9876", "LIB_FEAT_ID":197, "USAGE_TYPE": "HOME", "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-123-9876", "LIB_FEAT_ID":197}], {"FEAT_DESC": "818-333-7171", "LIB_FEAT_ID":198, "USAGE_TYPE": "MOBILE", "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-333-7171", "LIB_FEAT_ID":198}]}], "RECORD_SUMMARY": [{"DATA_SOURCE": "VIPS", "RECORD_COUNT":1}], "RECORDS": [{"DATA_SOURCE": "VIPS", "RECORD_ID": "XYZ234", "INTERNAL_ID":10, "MATCH_KEY": "", "MATCH_LEVEL_CODE": "", "ERRULE_CODE": ""}], "RELATED_ENTITIES": [{"ENTITY_ID":4, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+PNAME+ADDRESS-DOB", "ERRULE_CODE": "SFF", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Jane Donaldson", "RECORD_SUMMARY": [{"DATA_SOURCE": "PASSENGERS", "RECORD_COUNT":1}]}, {"ENTITY_ID":11, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+PHONE-DOB", "ERRULE_CODE": "SFF", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Martha Kent", "RECORD_SUMMARY": [{"DATA_SOURCE": "VIPS", "RECORD_COUNT":1}]}, {"ENTITY_ID":12, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+PHONE-DOB", "ERRULE_CODE": "SF1", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Kelly Rogers", "RECORD_SUMMARY": [{"DATA_SOURCE": "VIPS", "RECORD_COUNT":1}]}]}, {"RESOLVED_ENTITY": {"ENTITY_ID":11, "ENTITY_NAME": "Martha Kent", "FEATURES": {"ADDRESS": [{"FEAT_DESC": "888 Sepulveda Blvd, Los Angeles, CA 90034", "LIB_FEAT_ID":178, "FEAT_DESC_VALUES": [{"FEAT_DESC": "888 Sepulveda Blvd, Los Angeles, CA 90034", "LIB_FEAT_ID":178}]}], "DOB": [{"FEAT_DESC": "17-AUG-1978", "LIB_FEAT_ID":222, "FEAT_DESC_VALUES": [{"FEAT_DESC": "17-AUG-1978", "LIB_FEAT_ID":222}]}], "NAME": [{"FEAT_DESC": "Martha Kent", "LIB_FEAT_ID":221, "FEAT_DESC_VALUES": [{"FEAT_DESC": "Martha Kent", "LIB_FEAT_ID":221}]}], "PHONE": [{"FEAT_DESC": "818-123-9876", "LIB_FEAT_ID":197}], {"FEAT_DESC": "818-333-5757", "LIB_FEAT_ID":223, "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-333-5757", "LIB_FEAT_ID":223}]}]}, "RECORD_SUMMARY": [{"DATA_SOURCE": "VIPS", "RECORD_COUNT":1}], "RECORDS": [{"DATA_SOURCE": "VIPS", "RECORD_ID": "GHI123", "INTERNAL_ID":11, "MATCH_KEY": "", "MATCH_LEVEL_CODE": "", "ERRULE_CODE": ""}], "RELATED_ENTITIES": [{"ENTITY_ID":9, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+PNAME+ADDRESS-DOB", "ERRULE_CODE": "SFF", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Martha Wayne", "RECORD_SUMMARY": [{"DATA_SOURCE": "VIPS", "RECORD_COUNT":1}]}, {"ENTITY_ID":10, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+PHONE-DOB", "ERRULE_CODE": "SFF", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Jane Johnson", "RECORD_SUMMARY": [{"DATA_SOURCE": "VIPS", "RECORD_COUNT":1}]}]}, {"RESOLVED_ENTITY": {"ENTITY_ID":12, "ENTITY_NAME": "Kelly Rogers", "FEATURES": {"ADDRESS": [{"FEAT_DESC": "707 Seventh Ave, Los Angeles, CA 90043", "LIB_FEAT_ID":161, "FEAT_DESC_VALUES": [{"FEAT_DESC": "707 Seventh Ave, Los Angeles, CA 90043", "LIB_FEAT_ID":161}]}], "DOB": [{"FEAT_DESC": "15-JAN-1979", "LIB_FEAT_ID":236, "FEAT_DESC_VALUES": [{"FEAT_DESC": "15-JAN-1979", "LIB_FEAT_ID":236}]}], "NAME": [{"FEAT_DESC": "Kelly Rogers", "LIB_FEAT_ID":235, "FEAT_DESC_VALUES": [{"FEAT_DESC": "Kelly Rogers", "LIB_FEAT_ID":235}]}], "PHONE": [{"FEAT_DESC": "818-789-6543", "LIB_FEAT_ID":237, "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-789-6543", "LIB_FEAT_ID":237}]}]}]}

```

789-6543", "LIB_FEAT_ID":237}]}], {"FEAT_DESC": "818-333-7171", "LIB_FEAT_ID":198, "USAGE_TYPE": "MOBILE", "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-333-7171", "LIB_FEAT_ID":198}]}]}, "RECORD_SUMMARY": [{"DATA_SOURCE": "VIPS", "RECORD_COUNT":1}], "RECORDS": [{"DATA_SOURCE": "VIPS", "RECORD_ID": "JKL456", "INTERNAL_ID":12, "MATCH_KEY": "", "MATCH_LEVEL_CODE": "", "ERRULE_CODE": ""}], "RELATED_ENTITIES": [{"ENTITY_ID":8, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+ADDRESS-DOB", "ERRULE_CODE": "SFF", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Katrina Osmond", "RECORD_SUMMARY": [{"DATA_SOURCE": "EMPLOYEES", "RECORD_COUNT":1}]}, {"ENTITY_ID":10, "MATCH_LEVEL_CODE": "POSSIBLY RELATED", "MATCH_KEY": "+PHONE-DOB", "ERRULE_CODE": "SF1", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Jane Johnson", "RECORD_SUMMARY": [{"DATA_SOURCE": "VIPS", "RECORD_COUNT":1}]}]}, {"RESOLVED_ENTITY": {"ENTITY_ID":100002, "ENTITY_NAME": "Joe Schmoe", "FEATURES": {"EMAIL": [{"FEAT_DESC": "joeschmoe@nowhere.com", "LIB_FEAT_ID":100003, "FEAT_DESC_VALUES": [{"FEAT_DESC": "joeschmoe@nowhere.com", "LIB_FEAT_ID":100003}]}}}, "NAME": [{"FEAT_DESC": "Joe Schmoe", "LIB_FEAT_ID":100001, "FEAT_DESC_VALUES": [{"FEAT_DESC": "Joe Schmoe", "LIB_FEAT_ID":100001}]}}}, "PHONE": [{"FEAT_DESC": "702-555-1212", "LIB_FEAT_ID":100002, "FEAT_DESC_VALUES": [{"FEAT_DESC": "702-555-1212", "LIB_FEAT_ID":100002}]}}], "RECORD_SUMMARY": [{"DATA_SOURCE": "TEST", "RECORD_COUNT":1}], "RECORDS": [{"DATA_SOURCE": "TEST", "RECORD_ID": "ABC123", "INTERNAL_ID":100002, "MATCH_KEY": "", "MATCH_LEVEL_CODE": "", "ERRULE_CODE": ""}]}, "RELATED_ENTITIES": []}

```

Remarks

Used in conjunction with [FetchNext\(IntPtr\)](#) and [CloseExportReport\(IntPtr\)](#). Each [FetchNext\(IntPtr\)](#) call returns exported entity data as a JSON object.

WARNING: This method should only be used on systems containing less than a few million records. For larger systems, see

https://www.senzing.com/docs/tutorials/advanced_replication/.

The optionally specified bitwise-OR'd [SzFlag](#) values not only control how the operation is performed but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzExportFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzException](#)

If a failure occurs.

See Also

[SzExportDefaultFlags](#), [SzExportFlags](#), [FetchNext\(IntPtr\)](#), [CloseExportReport\(IntPtr\)](#),
[ExportCsvEntityReport\(string, SzFlag?\)](#)

FetchNext(IntPtr)

Fetches the next line of entity data from an open export report.

```
[SzConfigRetryable]
string FetchNext(IntPtr exportHandle)
```

Parameters

exportHandle [IntPtr](#)

The export handle for the export report from which to retrieve the next line of data.

Returns

[string](#)

The next line of export data whose format depends on which function was used to initiate the export, or `null` if there is no more data to be exported via the specified handle.

Examples

Usage (JSON Export):

```
// How to export entity data in JSON format
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // export the JSON data
    IntPtr exportHandle = engine.ExportJsonEntityReport(SzExportDefaultFlags);

    // read the data
    try
    {
```

```

        // fetch the first JSON record
        string jsonData = engine.FetchNext(exportHandle);

        while (jsonData != null)
        {
            // parse the JSON data
            JsonObject? jsonObject = JsonNode.Parse(jsonData)?.AsObject();

            // do something with the parsed data (varies by application)
            ProcessJsonRecord(jsonObject);

            // fetch the next JSON record
            jsonData = engine.FetchNext(exportHandle);
        }

    }

    finally
    {
        // close the export handle
        engine.CloseExportReport(exportHandle);
    }
}

}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to perform JSON export.", e);
}

```

Example JSON Line Result:

```

{"RESOLVED_ENTITY": {"ENTITY_ID": 1, "ENTITY_NAME": "Joseph Schmidt", "FEATURES": {"ADDRESS": [{"FEAT_DESC": "101 Main Street, Los Angeles, CA 90011", "LIB_FEAT_ID": 3, "FEAT_DESC_VALUES": [{"FEAT_DESC": "101 Main Street, Los Angeles, CA 90011", "LIB_FEAT_ID": 3}]}], "DOB": [{"FEAT_DESC": "12-JAN-1981", "LIB_FEAT_ID": 2, "FEAT_DESC_VALUES": [{"FEAT_DESC": "12-JAN-1981", "LIB_FEAT_ID": 2}]}], "NAME": [{"FEAT_DESC": "Joseph Schmidt", "LIB_FEAT_ID": 1, "FEAT_DESC_VALUES": [{"FEAT_DESC": "Joseph Schmidt", "LIB_FEAT_ID": 1}]}], "PHONE": [{"FEAT_DESC": "818-777-2424", "LIB_FEAT_ID": 4, "USAGE_TYPE": "HOME", "FEAT_DESC_VALUES": [{"FEAT_DESC": "818-777-2424", "LIB_FEAT_ID": 4}]}, {"FEAT_DESC": "213-555-1212", "LIB_FEAT_ID": 5, "USAGE_TYPE": "MOBILE", "FEAT_DESC_VALUES": [{"FEAT_DESC": "213-555-1212", "LIB_FEAT_ID": 5}]}]}, "RECORD_SUMMARY": [{"DATA_SOURCE": "PASSENGERS", "RECORD_COUNT": 1}], "RECORDS": [{"DATA_SOURCE": "PASSENGERS", "RECORD_ID": "ABC123", "INTERNAL_ID": 1, "MATCH_KEY": "", "MA

```

```

TCH_LEVEL_CODE:"", "ERRULE_CODE":""}]}}, "RELATED_ENTITIES": [{"ENTITY_ID":2, "MATCH_LEVEL_CODE":"POSSIBLY RELATED", "MATCH_KEY": "+PHONE-DOB", "ERRULE_CODE": "SF1", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Joann Smith", "RECORD_SUMMARY": [{"DATA_SOURCE": "PASSENGERS", "RECORD_COUNT":1}]}, {"ENTITY_ID":5, "MATCH_LEVEL_CODE":"POSSIBLY RELATED", "MATCH_KEY": "+ADDRESS-DOB", "ERRULE_CODE": "SFF", "IS_DISCLOSED":0, "IS_AMBIGUOUS":0, "ENTITY_NAME": "Bill Bandley", "RECORD_SUMMARY": [{"DATA_SOURCE": "EMPLOYEES", "RECORD_COUNT":1}]}]}
```

Usage (CSV Export):

```

// How to export entity data in CSV format
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // export the JSON data
    IntPtr exportHandle = engine.ExportCsvEntityReport("*", SzExportDefaultFlags);

    // read the data
    try
    {
        // fetch the CSV header line from the exported data
        string csvHeaders = engine.FetchNext(exportHandle);

        // process the CSV headers (varies by application)
        ProcessCsvHeaders(csvHeaders);

        // fetch the first CSV record from the exported data
        string csvRecord = engine.FetchNext(exportHandle);

        while (csvRecord != null)
        {
            // do something with the exported record (varies by application)
            ProcessCsvRecord(csvRecord);

            // fetch the next exported CSV record
            csvRecord = engine.FetchNext(exportHandle);
        }
    }
    finally
```

```

    {
        // close the export handle
        engine.CloseExportReport(exportHandle);
    }

}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to perform CSV export.", e);
}

```

Example CSV Header Result:

RESOLVED_ENTITY_ID,RESOLVED_ENTITY_NAME,RELATED_ENTITY_ID,MATCH_LEVEL,MATCH_LEVEL_CODE,MATCH_KEY,MATCH_KEY_DETAILS,IS_DISCLOSED,IS_AMBIGUOUS,DATA_SOURCE,RECORD_ID,JSON_DATA,FIRST_SEEN_DT,LAST_SEEN_DT,UNMAPPED_DATA,ERRULE_CODE,RELATED_ENTITY_NAME

Example CSV Data Result:

```

1,"Joseph Schmidt",0,0,"","","","",0,0,"PASSENGERS","ABC123",
{""RECORD_ID"":""ABC123"",""NAME_FIRST"":""Joseph"",""NAME_LAST"":""Schmidt"",""MOBILE_PHONE_NUMBER"":""213-555-1212"",""HOME_PHONE_NUMBER"":""818-777-2424"",""ADDR_FULL"":""101 Main Street, Los Angeles, CA 90011"",""DATE_OF_BIRTH"":""12-JAN-1981"",""DATA_SOURCE"":""PASSENGERS""}"}, "2025-10-17T23:21:01Z", "2025-10-17T23:21:01Z", "{}", "", ""

```

Remarks

Used in conjunction with [ExportJsonEntityReport\(SzFlag?\)](#), [ExportCsvEntityReport\(string, SzFlag?\)](#) and [CloseExportReport\(IntPtr\)](#).

If the export handle was obtained from [ExportCsvEntityReport\(string, SzFlag?\)](#), this returns the CSV header on the first call and exported entity data in CSV format on subsequent calls.

If the export handle was obtained from [ExportJsonEntityReport\(SzFlag?\)](#) this returns exported entity data as a JSON object.

When `null` is returned, the export report is complete and the caller should invoke [CloseExportReport\(IntPtr\)](#) to free resources.

Exceptions

[SzException](#)

If the specified export handle has already been [closed](#) or if any other failure occurs.

See Also

[CloseExportReport\(IntPtr\)](#), [ExportJsonEntityReport\(SzFlag?\)](#),
[ExportCsvEntityReport\(string, SzFlag?\)](#)

FindInterestingEntities(long, SzFlag?)

Experimental method.

```
[SzConfigRetryable]
string FindInterestingEntities(long entityID, SzFlag? flags = (SzFlag)0)
```

Parameters

entityID [long](#)

The entity ID identifying the entity that will be the focus for the interesting entities to be returned.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzFindInterestingEntitiesFlags](#) group control how the operation is performed and the content of the response. Omitting this parameter will default its value to [SzFindInterestingEntitiesDefaultFlags](#). Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) describing the interesting entities.

Examples

Usage:

```
// How to find interesting entities related to an entity via entity ID
try
{
```

```

// obtain the SzEnvironment (varies by application)
SzEnvironment env = GetEnvironment();

// get the engine
SzEngine engine = env.GetEngine();

// get the ID of an entity to retrieve (varies by application)
long entityID = GetEntityID();

// find the interesting entities by entity ID
string result = engine.FindInterestingEntities(
    entityID, SzFindInterestingEntitiesDefaultFlags);

// do something with the response JSON (varies by application)
JsonObject? jsonObject = JsonNode.Parse(result)?.AsObject();

. . .

}

catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Entity not found for entity ID.", e);

}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to find interesting entities by entity ID.", e);
}

```

Remarks

Contact Senzing support for the further information.

The optionally specified bitwise-OR'd [SzFlag](#) values may contain any flags, but only flags belonging to the [SzFindInterestingEntitiesFlags](#) group are guaranteed to be recognized (other [SzFlag](#) instances will be ignored unless they have equivalent bit flags to supported flags).

Exceptions

[SzNotFoundException](#)

If no entity could be found with the specified entity ID.

[SzException](#)

If a failure occurs.

See Also

[SzNoFlags](#), [SzFindInterestingEntitiesDefaultFlags](#), [SzFindInterestingEntitiesFlags](#)

FindInterestingEntities(string, string, SzFlag?)

Experimental method.

```
[SzConfigRetryable]
string FindInterestingEntities(string dataSourceCode, string recordID, SzFlag? flags
= (SzFlag)0)
```

Parameters

dataSourceCode [string](#)

The data source code that identifies the data source of the constituent record belonging to the entity that is the focus for the interesting entities to be returned.

recordID [string](#)

The record ID (unique within the scope of the record's data source) identifying the constituent record belonging to the entity that is the focus for the interesting entities to be returned.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzFindInterestingEntitiesFlags](#) group control how the operation is performed and the content of the response. Omitting this parameter will default its value to [SzFindInterestingEntitiesDefaultFlags](#). Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) describing the interesting entities.

Examples

Usage:

```
// How to find interesting entities related to an entity via record key
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // retrieve the entity by record key
    string result = engine.FindInterestingEntities(
        "TEST", "ABC123", SzFindInterestingEntitiesDefaultFlags);

    // do something with the response JSON (varies by application)
    JsonObject? jsonObject = JsonNode.Parse(result)?.AsObject();

    . .
}

}
catch (SzUnknownDataSourceException e)
{
    // handle the unknown data source exception
    LogError("Expected data source is not configured.", e);

}
catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Entity not found for record key.", e);

}
catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to find interesting entities by record key.", e);
}
```

Remarks

Contact Senzing support for the further information.

The optionally specified bitwise-OR'd [SzFlag](#) values may contain any flags, but only flags belonging to the [SzFindInterestingEntitiesFlags](#) group are guaranteed to be recognized

(other [SzFlag](#) instances will be ignored unless they have equivalent bit flags to supported flags).

Exceptions

[SzUnknownDataSourceException](#)

If an unrecognized data source code is specified.

[SzNotFoundException](#)

If no record can be found for the specified record ID.

[SzException](#)

If a failure occurs.

FindNetwork(ISet<long>, int, int, int, SzFlag?)

Retrieves a network of relationships among entities, specified by entity IDs.

```
[SzConfigRetryable]
string FindNetwork(ISet<long> entityIDs, int maxDegrees, int buildOutDegrees, int
buildOutMaxEntities, SzFlag? flags = SzFlag.SzEntityIncludeEntityName |
SzFlag.SzEntityIncludeRecordSummary | SzFlag.SzFindNetworkIncludeMatchingInfo)
```

Parameters

entityIDs [ISet<T>](#)<long>

The non-null [ISet<T>](#) of [long](#) entity ID's identifying the entities for which to build the network.

maxDegrees [int](#)

The maximum number of degrees for the path search between the specified entities. The maximum degrees of separation for the paths between entities must be specified so as to prevent the network growing beyond the desired size.

buildOutDegrees [int](#)

The number of relationship degrees to build out from each of the found entities on the network, or zero to prevent network build-out. If this is non-zero, the size of the network

can be limited to a maximum total number of build-out entities for the whole network.

buildOutMaxEntities [int](#)

The maximum number of entities to build out for the entire network. This limits the size of the build-out network when the build-out degrees is non-zero.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzFindNetworkFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter will default its value to the default recommended flags ([SzFindNetworkDefaultFlags](#)). Specifying `null` is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) describing the resultant entity network and the entities on the network.

Examples

Usage:

```
// How to find an entity network using entity ID's
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get entity ID's for the path endpoints (varies by application)
    ISet<long> entityIDs = GetNetworkEntityIDs();

    // determine the maximum path degrees (varies by application)
    int maxDegrees = 3;

    // determine the degrees to build-out the network (varies by application)
    int buildOutDegrees = 0;

    // determine the max entities to build-out (varies by application)
    int buildOutMaxEntities = 10;
```

```

// retrieve the entity network using the entity ID's
string result = engine.FindNetwork(entityIDs,
                                    maxDegrees,
                                    buildOutDegrees,
                                    buildOutMaxEntities,
                                    SzFindNetworkDefaultFlags);

// do something with the response JSON (varies by application)
JsonObject? jsonObject = JsonNode.Parse(result)?.AsObject();
JsonArray? pathArr = jsonObject?["ENTITY_PATHS"]?.AsArray();

for (int index = 0; pathArr != null && index < pathArr.Count; index++)
{
    JsonObject? path = pathArr[index]?.AsObject();

    long startEntityID = path?["START_ENTITY_ID"]?.GetValue<long>() ?? 0L;
    long endEntityID = path?["END_ENTITY_ID"]?.GetValue<long>() ?? 0L;
    JsonArray? pathIDs = path?["ENTITIES"]?.AsArray();

    for (int index2 = 0; pathIDs != null && index2 < pathIDs.Count; index2++)
    {
        long entityID = pathIDs[index2]?.GetValue<long>() ?? 0L;

        .
        .

    }
}

}

catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Entity not found for entity ID.", e);
}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to retrieve entity network.", e);
}

```

Example Result:

(formatted for readability)

```
{
  "ENTITY_PATHS": [
    {

```

```
"START_ENTITY_ID": 1,  
"END_ENTITY_ID": 7,  
"ENTITIES": [  
    1,  
    2,  
    6,  
    7  
],  
,  
{  
    "START_ENTITY_ID": 1,  
    "END_ENTITY_ID": 12,  
    "ENTITIES": [  
        1,  
        5,  
        8,  
        12  
    ],  
,  
{  
    "START_ENTITY_ID": 7,  
    "END_ENTITY_ID": 12,  
    "ENTITIES": []  
},  
]  
,  
"ENTITY_NETWORK_LINKS": [  
{  
    "MIN_ENTITY_ID": 1,  
    "MAX_ENTITY_ID": 2,  
    "MATCH_LEVEL_CODE": "POSSIBLY RELATED",  
    "MATCH_KEY": "\u002BPHONE-DOB",  
    "ERRULE_CODE": "SF1",  
    "IS_DISCLOSED": 0,  
    "IS_AMBIGUOUS": 0  
},  
{  
    "MIN_ENTITY_ID": 1,  
    "MAX_ENTITY_ID": 5,  
    "MATCH_LEVEL_CODE": "POSSIBLY RELATED",  
    "MATCH_KEY": "\u002BADDRESS-DOB",  
    "ERRULE_CODE": "SFF",  
    "IS_DISCLOSED": 0,  
    "IS_AMBIGUOUS": 0  
},  
{  
    "MIN_ENTITY_ID": 2,
```

```
"MAX_ENTITY_ID": 6,
"MATCH_LEVEL_CODE": "POSSIBLY RELATED",
"MATCH_KEY": "\u002BPHONE\u002BSURNAME-DOB",
"ERRULE_CODE": "CFF_SURNAME",
"IS_DISCLOSED": 0,
"IS_AMBIGUOUS": 0
},
{
"MIN_ENTITY_ID": 5,
"MAX_ENTITY_ID": 8,
"MATCH_LEVEL_CODE": "POSSIBLY RELATED",
"MATCH_KEY": "\u002BPHONE-DOB",
"ERRULE_CODE": "SF1",
"IS_DISCLOSED": 0,
"IS_AMBIGUOUS": 0
},
{
"MIN_ENTITY_ID": 6,
"MAX_ENTITY_ID": 7,
"MATCH_LEVEL_CODE": "POSSIBLY RELATED",
"MATCH_KEY": "\u002BADDRESS-DOB",
"ERRULE_CODE": "SFF",
"IS_DISCLOSED": 0,
"IS_AMBIGUOUS": 0
},
{
"MIN_ENTITY_ID": 8,
"MAX_ENTITY_ID": 12,
"MATCH_LEVEL_CODE": "POSSIBLY RELATED",
"MATCH_KEY": "\u002BADDRESS-DOB",
"ERRULE_CODE": "SFF",
"IS_DISCLOSED": 0,
"IS_AMBIGUOUS": 0
}
],
"ENTITIES": [
{
"RESOLVED_ENTITY": {
"ENTITY_ID": 1,
"ENTITY_NAME": "Joseph Schmidt",
"RECORD_SUMMARY": [
{
"DATA_SOURCE": "PASSENGERS",
"RECORD_COUNT": 1
}
]
}
```

```
        }
    },
{
  "RESOLVED_ENTITY": {
    "ENTITY_ID": 2,
    "ENTITY_NAME": "Joann Smith",
    "RECORD_SUMMARY": [
      {
        "DATA_SOURCE": "PASSENGERS",
        "RECORD_COUNT": 1
      }
    ]
  },
{
  "RESOLVED_ENTITY": {
    "ENTITY_ID": 5,
    "ENTITY_NAME": "Bill Bandley",
    "RECORD_SUMMARY": [
      {
        "DATA_SOURCE": "EMPLOYEES",
        "RECORD_COUNT": 1
      }
    ]
  },
{
  "RESOLVED_ENTITY": {
    "ENTITY_ID": 6,
    "ENTITY_NAME": "Craig Smith",
    "RECORD_SUMMARY": [
      {
        "DATA_SOURCE": "EMPLOYEES",
        "RECORD_COUNT": 1
      }
    ]
  },
{
  "RESOLVED_ENTITY": {
    "ENTITY_ID": 7,
    "ENTITY_NAME": "Kim Long",
    "RECORD_SUMMARY": [
      {
        "DATA_SOURCE": "EMPLOYEES",
        "RECORD_COUNT": 1
      }
    ]
  }
},
```

```

        }
    ]
},
{
    "RESOLVED_ENTITY": {
        "ENTITY_ID": 8,
        "ENTITY_NAME": "Katrina Osmond",
        "RECORD_SUMMARY": [
            {
                "DATA_SOURCE": "EMPLOYEES",
                "RECORD_COUNT": 1
            }
        ]
    },
    {
        "RESOLVED_ENTITY": {
            "ENTITY_ID": 12,
            "ENTITY_NAME": "Kelly Rogers",
            "RECORD_SUMMARY": [
                {
                    "DATA_SOURCE": "VIPS",
                    "RECORD_COUNT": 1
                }
            ]
        }
    }
]
}

```

Remarks

WARNING: Entity networks may be very large due to the volume of inter-related data in the repository. The parameters of this method can be used to limit the information returned.

The optionally specified bitwise-OR'd [SzFlag](#) values not only control how the operation is performed but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzFindNetworkFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzNotFoundException](#)

If any of the entities for the specified entity ID's cannot be found.

[SzException](#)

If a failure occurs.

See Also

[SzFindNetworkDefaultFlags](#), [SzFindNetworkFlags](#),
[FindNetwork\(ISet<\(string dataSourceCode, string recordID\)>, int, int, int, SzFlag?\)](#).

FindNetwork(ISet<(string dataSourceCode, string recordID)>, int, int, int, SzFlag?)

Retrieves a network of relationships among entities, specified by record IDs.

```
[SzConfigRetryable]
string FindNetwork(ISet<(string dataSourceCode, string recordID)> recordKeys, int
maxDegrees, int buildOutDegrees, int buildOutMaxEntities, SzFlag? flags =
SzFlag.SzEntityIncludeEntityName | SzFlag.SzEntityIncludeRecordSummary |
SzFlag.SzFindNetworkIncludeMatchingInfo)
```

Parameters

recordKeys [ISet<\(string dataSourceCode, string recordID\)>](#)

The non-null [ISet<T>](#) of tuples of data source code and record ID pairs identifying the constituent records of the entities for which to build the network.

maxDegrees [int](#)

The maximum number of degrees for the path search between the specified entities. The maximum degrees of separation for the paths between entities must be specified so as to prevent the network growing beyond the desired size.

buildOutDegrees [int](#)

The number of relationship degrees to build out from each of the found entities on the network, or zero to prevent network build-out. If this is non-zero, the size of the network can be limited to a maximum total number of build-out entities for the whole network.

`buildOutMaxEntities` [int](#)

The maximum number of entities to build out for the entire network. This limits the size of the build-out network when the build-out degrees is non-zero.

`flags` [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzFindNetworkFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter will default its value to the default recommended flags ([SzFindNetworkDefaultFlags](#)). Specifying `null` is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) describing the resultant entity network and the entities on the network.

Examples

Usage:

```
// How to find an entity network using record keys
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get entity ID's for the path endpoints (varies by application)
    ISet<(string, string)> recordKeys = GetNetworkRecordKeys();

    // determine the maximum path degrees (varies by application)
    int maxDegrees = 3;

    // determine the degrees to build-out the network (varies by application)
    int buildOutDegrees = 0;

    // determine the max entities to build-out (varies by application)
    int buildOutMaxEntities = 10;

    // retrieve the entity network using the record keys
    string result = engine.FindNetwork(recordKeys,
```

```

        maxDegrees,
        buildOutDegrees,
        buildOutMaxEntities,
        SzFindNetworkDefaultFlags);

// do something with the response JSON (varies by application)
JsonObject? jsonObject = JsonNode.Parse(result)?.AsObject();
JsonArray? pathArr = jsonObject?["ENTITY_PATHS"]?.AsArray();

for (int index = 0; pathArr != null && index < pathArr.Count; index++)
{
    JsonObject? path = pathArr[index]?.AsObject();

    long startEntityID = path?["START_ENTITY_ID"]?.GetValue<long>() ?? 0L;
    long endEntityID = path?["END_ENTITY_ID"]?.GetValue<long>() ?? 0L;
    JsonArray? pathIDs = path?["ENTITIES"]?.AsArray();

    for (int index2 = 0; pathIDs != null && index2 < pathIDs.Count; index2++)
    {
        long entityID = pathIDs[index2]?.GetValue<long>() ?? 0L;

        . . .
    }
}

}

catch (SzUnknownDataSourceException e)
{
    // handle the unknown data source exception
    LogError("Expected data source is not configured.", e);
}

catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Entity not found for record key.", e);
}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to retrieve entity network.", e);
}

```

Example Result: (formatted for readability)

```
{
  "ENTITY_PATHS": [
    {
      "START_ENTITY_ID": 1,
      "END_ENTITY_ID": 7,
      "ENTITIES": [
        1,
        2,
        6,
        7
      ]
    },
    {
      "START_ENTITY_ID": 1,
      "END_ENTITY_ID": 12,
      "ENTITIES": [
        1,
        5,
        8,
        12
      ]
    },
    {
      "START_ENTITY_ID": 7,
      "END_ENTITY_ID": 12,
      "ENTITIES": []
    }
  ],
  "ENTITY_NETWORK_LINKS": [
    {
      "MIN_ENTITY_ID": 1,
      "MAX_ENTITY_ID": 2,
      "MATCH_LEVEL_CODE": "POSSIBLY RELATED",
      "MATCH_KEY": "\u002BPHONE-DOB",
      "ERRULE_CODE": "SF1",
      "IS_DISCLOSED": 0,
      "IS_AMBIGUOUS": 0
    },
    {
      "MIN_ENTITY_ID": 1,
      "MAX_ENTITY_ID": 5,
      "MATCH_LEVEL_CODE": "POSSIBLY RELATED",
      "MATCH_KEY": "\u002BADDRESS-DOB",
      "ERRULE_CODE": "SFF",
      "IS_DISCLOSED": 0,
      "IS_AMBIGUOUS": 0
    }
  ]
}
```

```
},
{
  "MIN_ENTITY_ID": 2,
  "MAX_ENTITY_ID": 6,
  "MATCH_LEVEL_CODE": "POSSIBLY RELATED",
  "MATCH_KEY": "\u002BPHONE\u002BSURNAME-DOB",
  "ERRULE_CODE": "CFF_SURNAME",
  "IS_DISCLOSED": 0,
  "IS_AMBIGUOUS": 0
},
{
  "MIN_ENTITY_ID": 5,
  "MAX_ENTITY_ID": 8,
  "MATCH_LEVEL_CODE": "POSSIBLY RELATED",
  "MATCH_KEY": "\u002BPHONE-DOB",
  "ERRULE_CODE": "SF1",
  "IS_DISCLOSED": 0,
  "IS_AMBIGUOUS": 0
},
{
  "MIN_ENTITY_ID": 6,
  "MAX_ENTITY_ID": 7,
  "MATCH_LEVEL_CODE": "POSSIBLY RELATED",
  "MATCH_KEY": "\u002BADDRESS-DOB",
  "ERRULE_CODE": "SFF",
  "IS_DISCLOSED": 0,
  "IS_AMBIGUOUS": 0
},
{
  "MIN_ENTITY_ID": 8,
  "MAX_ENTITY_ID": 12,
  "MATCH_LEVEL_CODE": "POSSIBLY RELATED",
  "MATCH_KEY": "\u002BADDRESS-DOB",
  "ERRULE_CODE": "SFF",
  "IS_DISCLOSED": 0,
  "IS_AMBIGUOUS": 0
}
],
"ENTITIES": [
  {
    "RESOLVED_ENTITY": {
      "ENTITY_ID": 1,
      "ENTITY_NAME": "Joseph Schmidt",
      "RECORD_SUMMARY": [
        {
          "DATA_SOURCE": "PASSENGERS",

```

```
        "RECORD_COUNT": 1
    }
]
}
},
{
"RESOLVED_ENTITY": {
    "ENTITY_ID": 2,
    "ENTITY_NAME": "Joann Smith",
    "RECORD_SUMMARY": [
        {
            "DATA_SOURCE": "PASSENGERS",
            "RECORD_COUNT": 1
        }
    ]
},
{
"RESOLVED_ENTITY": {
    "ENTITY_ID": 5,
    "ENTITY_NAME": "Bill Bandley",
    "RECORD_SUMMARY": [
        {
            "DATA_SOURCE": "EMPLOYEES",
            "RECORD_COUNT": 1
        }
    ]
},
{
"RESOLVED_ENTITY": {
    "ENTITY_ID": 6,
    "ENTITY_NAME": "Craig Smith",
    "RECORD_SUMMARY": [
        {
            "DATA_SOURCE": "EMPLOYEES",
            "RECORD_COUNT": 1
        }
    ]
},
{
"RESOLVED_ENTITY": {
    "ENTITY_ID": 7,
    "ENTITY_NAME": "Kim Long",
    "RECORD_SUMMARY": [
```

```

        {
            "DATA_SOURCE": "EMPLOYEES",
            "RECORD_COUNT": 1
        }
    ]
}
},
{
    "RESOLVED_ENTITY": {
        "ENTITY_ID": 8,
        "ENTITY_NAME": "Katrina Osmond",
        "RECORD_SUMMARY": [
            {
                "DATA_SOURCE": "EMPLOYEES",
                "RECORD_COUNT": 1
            }
        ]
    }
},
{
    "RESOLVED_ENTITY": {
        "ENTITY_ID": 12,
        "ENTITY_NAME": "Kelly Rogers",
        "RECORD_SUMMARY": [
            {
                "DATA_SOURCE": "VIPS",
                "RECORD_COUNT": 1
            }
        ]
    }
}
]
}

```

Remarks

WARNING: Entity networks may be very large due to the volume of inter-related data in the repository. The parameters of this method can be used to limit the information returned.

The optionally specified bitwise-OR'd [SzFlag](#) values not only control how the operation is performed but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzFindNetworkFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzUnknownDataSourceException](#)

If an unrecognized data source code is specified.

[SzNotFoundException](#)

If any of the records for the specified data source code and record ID pairs cannot be found.

[SzException](#)

If a failure occurs.

See Also

[SzFindNetworkDefaultFlags](#), [SzFindNetworkFlags](#),
[FindNetwork\(ISet<long>, int, int, int, SzFlag?\)](#).

FindPath(long, long, int, ISet<long>, ISet<string>, SzFlag?)

Searches for the shortest relationship path between two entities, specified by entity IDs.

```
[SzConfigRetryable]
string FindPath(long startEntityID, long endEntityID, int maxDegrees, ISet<long>
avoidEntityIDs = null, ISet<string> requiredDataSources = null, SzFlag? flags =
SzFlag.SzEntityIncludeEntityName | SzFlag.SzEntityIncludeRecordSummary |
SzFlag.SzFindPathIncludeMatchingInfo)
```

Parameters

[startEntityID](#) [long](#)

The entity ID of the first entity.

[endEntityID](#) [long](#)

The entity ID of the second entity.

[maxDegrees](#) [int](#)

The maximum number of degrees for the path search.

`avoidEntityIDs` [ISet<long>](#)

The optional [ISet<T>](#) of `long` entity ID's identifying entities to be avoided when finding the path, or `null` if no entities are to be avoided. By default the entities will be avoided unless necessary to find the path. To strictly avoid the entities specify the [SzFindPath StrictAvoid](#) flag.

`requiredDataSources` [ISet<string>](#)

The optional [ISet<T>](#) of non-null `string` data source codes identifying the data sources for which at least one record must be included on the path, or `null` if none are required.

`flags` [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzFindPathFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter will default its value to the default recommended flags ([SzFindPathDefaultFlags](#)).

Specifying `null` is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON `string` describing the resultant entity path which may be an empty path if no path exists between the two entities given the path parameters.

Examples

Usage:

```
// How to find an entity path using entity ID's
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get entity ID's for the path endpoints (varies by application)
    long startEntityID = GetPathStartEntityID();
    long endEntityID = GetPathEndEntityID();

    // determine the maximum path degrees (varies by application)
```

```

int maxDegrees = 4;

// determine any entities to be avoided (varies by application)
ISet<long> avoidEntities = GetPathAvoidEntityIDs();

// determine any data sources to require in the path (varies by application)
ISet<string>? requiredSources = null;

// retrieve the entity path using the entity ID's
string result = engine.FindPath(startEntityID,
                                 endEntityID,
                                 maxDegrees,
                                 avoidEntities,
                                 requiredSources,
                                 SzFindPathDefaultFlags);

// do something with the response JSON (varies by application)
JsonObject? jsonObject = JsonNode.Parse(result)?.AsObject();
JsonArray? pathArr = jsonObject?["ENTITY_PATHS"]?.AsArray();

for (int index = 0; pathArr != null && index < pathArr.Count; index++)
{
    JsonObject? path = pathArr[index]?.AsObject();
    JsonArray? entityIDs = path?["ENTITIES"]?.AsArray();

    for (int index2 = 0; entityIDs != null && index2 <
entityIDs.Count; index2++)
    {
        long entityID = entityIDs[index2]?.GetValue<long>() ?? 0L;

        . . .
    }
}

}

catch (SzUnknownDataSourceException e)
{
    // handle the unknown data source exception
    LogError("Expected data source is not configured.", e);
}

catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Entity not found for entity ID.", e);
}

```

```

}
catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to retrieve entity path by entity ID.", e);
}

```

Example Result: (formatted for readability)

```

{
    "ENTITY_PATHS": [
        {
            "START_ENTITY_ID": 1,
            "END_ENTITY_ID": 12,
            "ENTITIES": [
                1,
                5,
                8,
                12
            ]
        }
    ],
    "ENTITY_PATH_LINKS": [
        {
            "MIN_ENTITY_ID": 1,
            "MAX_ENTITY_ID": 5,
            "MATCH_LEVEL_CODE": "POSSIBLY RELATED",
            "MATCH_KEY": "\u002BADDRESS-DOB",
            "ERRULE_CODE": "SFF",
            "IS_DISCLOSED": 0,
            "IS_AMBIGUOUS": 0
        },
        {
            "MIN_ENTITY_ID": 5,
            "MAX_ENTITY_ID": 8,
            "MATCH_LEVEL_CODE": "POSSIBLY RELATED",
            "MATCH_KEY": "\u002BPHONE-DOB",
            "ERRULE_CODE": "SF1",
            "IS_DISCLOSED": 0,
            "IS_AMBIGUOUS": 0
        },
        {
            "MIN_ENTITY_ID": 8,
            "MAX_ENTITY_ID": 12,
            "MATCH_LEVEL_CODE": "POSSIBLY RELATED",

```

```
"MATCH_KEY": "\u002BADDRESS-DOB",
"ERRULE_CODE": "SFF",
"IS_DISCLOSED": 0,
"IS_AMBIGUOUS": 0
},
],
"ENTITIES": [
{
  "RESOLVED_ENTITY": {
    "ENTITY_ID": 1,
    "ENTITY_NAME": "Joseph Schmidt",
    "RECORD_SUMMARY": [
      {
        "DATA_SOURCE": "PASSENGERS",
        "RECORD_COUNT": 1
      }
    ]
  }
},
{
  "RESOLVED_ENTITY": {
    "ENTITY_ID": 5,
    "ENTITY_NAME": "Bill Bandley",
    "RECORD_SUMMARY": [
      {
        "DATA_SOURCE": "EMPLOYEES",
        "RECORD_COUNT": 1
      }
    ]
  }
},
{
  "RESOLVED_ENTITY": {
    "ENTITY_ID": 8,
    "ENTITY_NAME": "Katrina Osmond",
    "RECORD_SUMMARY": [
      {
        "DATA_SOURCE": "EMPLOYEES",
        "RECORD_COUNT": 1
      }
    ]
  }
},
{
  "RESOLVED_ENTITY": {
    "ENTITY_ID": 12,
```

```
"ENTITY_NAME": "Kelly Rogers",
"RECORD_SUMMARY": [
    {
        "DATA_SOURCE": "VIPS",
        "RECORD_COUNT": 1
    }
]
}
]
```

Remarks

The returned path is the shortest path among the paths that satisfy the parameters.

The optionally specified bitwise-OR'd [SzFlag](#) values not only control how the operation is performed but also the level of detail provided for the entity path and those entities on the path. Any [SzFlag](#) value may be included, but only flags belonging to the [SzFindPathFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzNotFoundException](#)

If either the path-start or path-end entities for the specified entity ID's cannot be found.

[SzUnknownDataSourceException](#)

If an unrecognized required data source is specified.

[SzException](#)

If a failure occurs.

See Also

[SzFindPathDefaultFlags](#), [SzFindPathFlags](#),

[FindPath\(string, string, string, string, int, ISet<\(string dataSourceCode, string recordID\)>, ISet<string>, SzFlag?\)](#)

[FindPath\(string, string, string, string, int, ISet<\(string dataSourceCode, string recordID\)>, ISet<string>,](#)

SzFlag?)

Searches for the shortest relationship path between two entities, specified by record IDs.

```
[SzConfigRetryable]
string FindPath(string startDataSourceCode, string startRecordID, string
endDataSourceCode, string endRecordID, int maxDegrees, ISet<(string dataSourceCode,
string recordID)> avoidRecordKeys = null, ISet<string> requiredDataSources = null,
SzFlag? flags = SzFlag.SzEntityIncludeEntityName |
SzFlag.SzEntityIncludeRecordSummary | SzFlag.SzFindPathIncludeMatchingInfo)
```

Parameters

startDataSourceCode [string](#)

The data source code identifying the data source for the starting record of the requested path.

startRecordID [string](#)

The record ID that uniquely identifies the starting record of the requested path within the scope of its associated data source.

endDataSourceCode [string](#)

The data source code identifying the data source for the ending record of the requested path.

endRecordID [string](#)

The record ID that uniquely identifies the ending record of the requested path within the scope of its associated data source.

maxDegrees [int](#)

The maximum number of degrees for the path search.

avoidRecordKeys [ISet](#)<(string dataSourceCode, string recordID)>

The optional [ISet<T>](#) of tuples containing data source code and record ID pairs identifying records whose entities are to be avoided when finding the path, or [null](#) if no entities are to be avoided. By default the entities will be avoided unless necessary to find the path. To strictly avoid the entities specify the [SzFindPathStrictAvoid](#) flag.

`requiredDataSources` [ISet<string>](#)

The optional [ISet<T>](#) of non-null `string` data source codes identifying the data sources for which at least one record must be included on the path, or `null` if none are required.

`flags` [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzFindPathFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter will default its value to the default recommended flags ([SzFindPathDefaultFlags](#)).

Specifying `null` is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON `string` describing the resultant entity path which may be an empty path if no path exists between the two entities given the path parameters.

Examples

Usage:

```
// How to find an entity path using record keys
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get entity ID's for the path endpoints (varies by application)
    (string dataSourceCode, string recordID) startRecordKey
= GetPathStartRecordKey();
    (string dataSourceCode, string recordID) endRecordKey = GetPathEndRecordKey();

    // determine the maximum path degrees (varies by application)
    int maxDegrees = 4;

    // determine any records to be avoided (varies by application)
    ISet<(string, string)> avoidRecords = GetPathAvoidRecordKeys();

    // determine any data sources to require in the path (varies by application)
    ISet<string>? requiredSources = null;
```

```

// retrieve the entity path using the record keys
string result = engine.FindPath(startRecordKey.dataSourceCode,
                                startRecordKey.recordID,
                                endRecordKey.dataSourceCode,
                                endRecordKey.recordID,
                                maxDegrees,
                                avoidRecords,
                                requiredSources,
                                SzFindPathDefaultFlags);

// do something with the response JSON (varies by application)
JsonObject? jsonObject = JsonNode.Parse(result)?.AsObject();
JsonArray? pathArr = jsonObject?["ENTITY_PATHS"]?.AsArray();

for (int index = 0; pathArr != null && index < pathArr.Count; index++)
{
    JsonObject? path = pathArr[index]?.AsObject();
    JsonArray? entityIDs = path?["ENTITIES"]?.AsArray();

        for (int index2 = 0; entityIDs != null && index2 <
entityIDs.Count; index2++)
    {
        long entityID = entityIDs[index2]?.GetValue<long>() ?? 0L;

        . . .

    }
}

}

catch (SzUnknownDataSourceException e)
{
    // handle the unknown data source exception
    LogError("Expected data source is not configured.", e);
}

catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Entity not found for record key.", e);
}

catch (SzException e)
{
    // handle or rethrow the other exceptions
}

```

```
        LogError("Failed to retrieve entity path by record key.", e);
    }
```

Example Result: (formatted for readability)

```
{
  "ENTITY_PATHS": [
    {
      "START_ENTITY_ID": 1,
      "END_ENTITY_ID": 12,
      "ENTITIES": [
        1,
        5,
        8,
        12
      ]
    }
  ],
  "ENTITY_PATH_LINKS": [
    {
      "MIN_ENTITY_ID": 1,
      "MAX_ENTITY_ID": 5,
      "MATCH_LEVEL_CODE": "POSSIBLY RELATED",
      "MATCH_KEY": "\u002BADDRESS-DOB",
      "ERRULE_CODE": "SFF",
      "IS_DISCLOSED": 0,
      "IS_AMBIGUOUS": 0
    },
    {
      "MIN_ENTITY_ID": 5,
      "MAX_ENTITY_ID": 8,
      "MATCH_LEVEL_CODE": "POSSIBLY RELATED",
      "MATCH_KEY": "\u002BPHONE-DOB",
      "ERRULE_CODE": "SF1",
      "IS_DISCLOSED": 0,
      "IS_AMBIGUOUS": 0
    },
    {
      "MIN_ENTITY_ID": 8,
      "MAX_ENTITY_ID": 12,
      "MATCH_LEVEL_CODE": "POSSIBLY RELATED",
      "MATCH_KEY": "\u002BADDRESS-DOB",
      "ERRULE_CODE": "SFF",
      "IS_DISCLOSED": 0,
      "IS_AMBIGUOUS": 0
    }
  ]
}
```

```
        },
    ],
    "ENTITIES": [
        {
            "RESOLVED_ENTITY": {
                "ENTITY_ID": 1,
                "ENTITY_NAME": "Joseph Schmidt",
                "RECORD_SUMMARY": [
                    {
                        "DATA_SOURCE": "PASSENGERS",
                        "RECORD_COUNT": 1
                    }
                ]
            }
        },
        {
            "RESOLVED_ENTITY": {
                "ENTITY_ID": 5,
                "ENTITY_NAME": "Bill Bandley",
                "RECORD_SUMMARY": [
                    {
                        "DATA_SOURCE": "EMPLOYEES",
                        "RECORD_COUNT": 1
                    }
                ]
            }
        },
        {
            "RESOLVED_ENTITY": {
                "ENTITY_ID": 8,
                "ENTITY_NAME": "Katrina Osmond",
                "RECORD_SUMMARY": [
                    {
                        "DATA_SOURCE": "EMPLOYEES",
                        "RECORD_COUNT": 1
                    }
                ]
            }
        },
        {
            "RESOLVED_ENTITY": {
                "ENTITY_ID": 12,
                "ENTITY_NAME": "Kelly Rogers",
                "RECORD_SUMMARY": [
                    {
                        "DATA_SOURCE": "VIPS",
                        "RECORD_COUNT": 1
                    }
                ]
            }
        }
    ]
}
```

```
        "RECORD_COUNT": 1
    }
]
}
]
}
```

Remarks

The returned path is the shortest path among the paths that satisfy the parameters.

The optionally specified bitwise-OR'd [SzFlag](#) values may contain any [SzFlag](#) value, but only flags belonging to the [SzFindPathFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags).

Exceptions

[SzNotFoundException](#)

If either the path-start or path-end records for the specified data source code and record ID pairs cannot be found.

[SzUnknownDataSourceException](#)

If an unrecognized data source code is specified.

[SzException](#)

If a failure occurs.

See Also

[SzFindPathDefaultFlags](#), [SzFindPathFlags](#),
[FindPath\(long, long, int, ISet<long>, ISet<string>, SzFlag?\)](#).

GetEntity(long, SzFlag?)

Retrieves information about an entity, specified by entity ID.

```
[SzConfigRetryable]
string GetEntity(long entityID, SzFlag? flags =
SzFlag.SzEntityIncludePossiblySameRelations |
SzFlag.SzEntityIncludePossiblyRelatedRelations |
```

```
SzFlag.SzEntityIncludeNameOnlyRelations | SzFlag.SzEntityIncludeDisclosedRelations |  
SzFlag.SzEntityIncludeRepresentativeFeatures | SzFlag.SzEntityIncludeEntityName |  
SzFlag.SzEntityIncludeRecordSummary | SzFlag.SzEntityIncludeRecordData |  
SzFlag.SzEntityIncludeRecordMatchingInfo | SzFlag.SzEntityIncludeRelatedEntityName |  
SzFlag.SzEntityIncludeRelatedMatchingInfo |  
SzFlag.SzEntityIncludeRelatedRecordSummary)
```

Parameters

entityID [long](#)

The entity ID identifying the entity to retrieve.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzEntityFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter defaults it value to [SzEntityDefaultFlags](#) for the default recommended flags. Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) describing the entity.

Examples

Usage:

```
// How to retrieve an entity via its entity ID  
try  
{  
    // obtain the SzEnvironment (varies by application)  
    SzEnvironment env = GetEnvironment();  
  
    // get the engine  
    SzEngine engine = env.GetEngine();  
  
    // get the ID of an entity to retrieve (varies by application)  
    long entityID = GetEntityID();  
  
    // retrieve the entity by entity ID  
    string result = engine.GetEntity(entityID, SzEntityDefaultFlags);
```

```

// do something with the response JSON (varies by application)
JsonObject? jsonObject = JsonNode.Parse(result)?.AsObject();
JsonObject? entity = jsonObject?["RESOLVED_ENTITY"]?.AsObject();
string? entityName = entity?["ENTITY_NAME"]?.GetValue<string>();

. . .

if (jsonObject != null && jsonObject.ContainsKey("RECORDS"))
{
    JSONArray? recordArr = jsonObject["RECORDS"]?.AsArray();

    for (int index = 0; recordArr != null && index < recordArr.Count; index++)
    {
        JsonObject? record = recordArr[index]?.AsObject();

        string? dataSource = record?["DATA_SOURCE"]?.GetValue<string>();
        string? recordID = record?["RECORD_ID"]?.GetValue<string>();

        . . .
    }
}

}

catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Entity not found for entity ID.", e);
}

}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to retrieve entity by entity ID.", e);
}

```

Example Result: (formatted for readability)

```
{
  "RESOLVED_ENTITY": {
    "ENTITY_ID": 100002,
    "ENTITY_NAME": "Joe Schmoe",
    "FEATURES": {
      "EMAIL": [

```

```
{  
    "FEAT_DESC": "joeschmoe@nowhere.com",  
    "LIB_FEAT_ID": 100003,  
    "FEAT_DESC_VALUES": [  
        {  
            "FEAT_DESC": "joeschmoe@nowhere.com",  
            "LIB_FEAT_ID": 100003  
        }  
    ]  
},  
"NAME": [  
    {  
        "FEAT_DESC": "Joe Schmoe",  
        "LIB_FEAT_ID": 100001,  
        "FEAT_DESC_VALUES": [  
            {  
                "FEAT_DESC": "Joe Schmoe",  
                "LIB_FEAT_ID": 100001  
            }  
        ]  
    }  
],  
"PHONE": [  
    {  
        "FEAT_DESC": "702-555-1212",  
        "LIB_FEAT_ID": 100002,  
        "FEAT_DESC_VALUES": [  
            {  
                "FEAT_DESC": "702-555-1212",  
                "LIB_FEAT_ID": 100002  
            }  
        ]  
    }  
],  
},  
"RECORD_SUMMARY": [  
    {  
        "DATA_SOURCE": "TEST",  
        "RECORD_COUNT": 1  
    }  
],  
"RECORDS": [  
    {  
        "DATA_SOURCE": "TEST",  
        "RECORD_ID": "ABC123",  
        "RECORD_TYPE": "TEST",  
        "RECORD_VALUE": "TEST",  
        "RECORD_TIMESTAMP": "2023-10-01T12:00:00Z"  
    }  
]
```

```

    "INTERNAL_ID": 100002,
    "MATCH_KEY": "",
    "MATCH_LEVEL_CODE": "",
    "ERRULE_CODE": ""

}
]
},
"RELATED_ENTITIES": []
}

```

Remarks

The optionally specified bitwise-OR'd [SzFlag](#) values not only controls how the operation is performed, but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzEntityFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzNotFoundException](#)

If no entity could be found with the specified entity ID.

[SzException](#)

If a failure occurs.

See Also

[SzEntityDefaultFlags](#), [SzEntityBriefDefaultFlags](#), [SzEntityFlags](#)

GetEntity(string, string, SzFlag?)

Retrieves information about an entity, specified by record ID.

```
[SzConfigRetryable]
string GetEntity(string dataSourceCode, string recordID, SzFlag? flags =
SzFlag.SzEntityIncludePossiblySameRelations |
SzFlag.SzEntityIncludePossiblyRelatedRelations |
SzFlag.SzEntityIncludeNameOnlyRelations | SzFlag.SzEntityIncludeDisclosedRelations |
SzFlag.SzEntityIncludeRepresentativeFeatures | SzFlag.SzEntityIncludeEntityName |
SzFlag.SzEntityIncludeRecordSummary | SzFlag.SzEntityIncludeRecordData |
SzFlag.SzEntityIncludeRecordMatchingInfo | SzFlag.SzEntityIncludeRelatedEntityName |
```

```
SzFlag.SzEntityIncludeRelatedMatchingInfo |  
SzFlag.SzEntityIncludeRelatedRecordSummary)
```

Parameters

dataSourceCode [string](#)

The data source code that identifies the data source of the constituent record belonging to the entity to be retrieved.

recordID [string](#)

The record ID (unique within the scope of the record's data source) identifying the constituent record belonging to the entity to be retrieved.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzEntityFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter defaults its value to [SzEntityDefaultFlags](#) for the default recommended flags. Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) describing the entity.

Examples

Usage:

```
// How to retrieve an entity via its record key  
try  
{  
    // obtain the SzEnvironment (varies by application)  
    SzEnvironment env = GetEnvironment();  
  
    // get the engine  
    SzEngine engine = env.GetEngine();  
  
    // retrieve the entity by record key  
    string result = engine.GetEntity("TEST", "ABC123", SzEntityDefaultFlags);
```

```

// do something with the response JSON (varies by application)
JsonObject? jsonObject = JsonNode.Parse(result)?.AsObject();
JsonObject? entity = jsonObject?["RESOLVED_ENTITY"]?.AsObject();
string? entityName = entity?["ENTITY_NAME"]?.GetValue<string>();

. . .

if (jsonObject != null && jsonObject.ContainsKey("RECORDS"))
{
    JSONArray? recordArr = jsonObject["RECORDS"]?.AsArray();

    for (int index = 0; recordArr != null && index < recordArr.Count; index++)
    {
        JsonObject? record = recordArr[index]?.AsObject();

        string? dataSource = record?["DATA_SOURCE"]?.GetValue<string>();
        string? recordID = record?["RECORD_ID"]?.GetValue<string>();

        . . .
    }
}

}

catch (SzUnknownDataSourceException e)
{
    // handle the unknown data source exception
    LogError("Expected data source is not configured.", e);
}

catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Entity not found for record key.", e);
}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to retrieve entity by record key.", e);
}

```

Example Result: (formatted for readability)

```
{  
  "RESOLVED_ENTITY": {  
    "ENTITY_ID": 100002,  
    "ENTITY_NAME": "Joe Schmoe",  
    "FEATURES": {  
      "EMAIL": [  
        {  
          "FEAT_DESC": "joeschmoe@nowhere.com",  
          "LIB_FEAT_ID": 100003,  
          "FEAT_DESC_VALUES": [  
            {  
              "FEAT_DESC": "joeschmoe@nowhere.com",  
              "LIB_FEAT_ID": 100003  
            }  
          ]  
        }  
      ],  
      "NAME": [  
        {  
          "FEAT_DESC": "Joe Schmoe",  
          "LIB_FEAT_ID": 100001,  
          "FEAT_DESC_VALUES": [  
            {  
              "FEAT_DESC": "Joe Schmoe",  
              "LIB_FEAT_ID": 100001  
            }  
          ]  
        }  
      ],  
      "PHONE": [  
        {  
          "FEAT_DESC": "702-555-1212",  
          "LIB_FEAT_ID": 100002,  
          "FEAT_DESC_VALUES": [  
            {  
              "FEAT_DESC": "702-555-1212",  
              "LIB_FEAT_ID": 100002  
            }  
          ]  
        }  
      ]  
    },  
    "RECORD_SUMMARY": [  
      {  
        "DATA_SOURCE": "TEST",  
        "RECORD_COUNT": 1  
      }  
    ]  
  }  
}
```

```
        }
    ],
    "RECORDS": [
        {
            "DATA_SOURCE": "TEST",
            "RECORD_ID": "ABC123",
            "INTERNAL_ID": 100002,
            "MATCH_KEY": "",
            "MATCH_LEVEL_CODE": "",
            "ERRULE_CODE": ""
        }
    ]
},
"RELATED_ENTITIES": []
}
```

Remarks

The optionally specified bitwise-OR'd [SzFlag](#) values not only controls how the operation is performed, but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzEntityFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzUnknownDataSourceException](#)

If an unrecognized data source code is specified.

[SzNotFoundException](#)

If no entity could be found with the specified entity ID.

[SzException](#)

If a failure occurs.

See Also

[SzEntityDefaultFlags](#), [SzEntityBriefDefaultFlags](#), [SzEntityFlags](#)

GetRecord(string, string, SzFlag?)

Retrieves information about a record.

```
[SzConfigRetryable]
string GetRecord(string dataSourceCode, string recordID, SzFlag? flags =
SzFlag.SzEntityIncludeRecordJsonData)
```

Parameters

dataSourceCode [string](#)

The data source code identifying the data source for the record.

recordID [string](#)

The record ID that uniquely identifies the record within the scope of its associated data source.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzRecordFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter defaults it value to [SzRecordDefaultFlags](#) for the default recommended flags. Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) describing the record.

Examples

Usage:

```
// How to retrieve a record via its record key
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // retrieve the entity by record key
    string result = engine.GetRecord("TEST", "ABC123", SzRecordDefaultFlags);
```

```

// do something with the response JSON (varies by application)
JsonObject? jsonObject = JsonNode.Parse(result)?.AsObject();
string? dataSource = jsonObject?["DATA_SOURCE"]?.GetValue<string>();
string? recordID = jsonObject?["RECORD_ID"]?.GetValue<string>();

. . .

}

catch (SzUnknownDataSourceException e)
{
    // handle the unknown data source exception
    LogError("Expected data source is not configured.", e);

}

catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Record not found for record key.", e);

}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to retrieve record by record key.", e);
}

```

Example Result:

(formatted for readability)

```
{
    "DATA_SOURCE": "TEST",
    "RECORD_ID": "ABC123",
    "JSON_DATA": {
        "DATA_SOURCE": "TEST",
        "RECORD_ID": "ABC123",
        "NAME_FULL": "Joe Schmoe",
        "PHONE_NUMBER": "702-555-1212",
        "EMAIL_ADDRESS": "joeschmoe@nowhere.com"
    }
}
```

Remarks

The returned information contains the original record data that was loaded and may contain other information depending on the flags parameter.

The optionally specified bitwise-OR'd [SzFlag](#) values not only control how the operation is performed but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzRecordFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzUnknownDataSourceException](#)

If an unrecognized data source code is specified.

[SzNotFoundException](#)

If the record for the specified data source code and record ID pairs cannot be found.

[SzException](#)

If a failure occurs.

See Also

[SzRecordDefaultFlags](#), [SzRecordFlags](#)

GetRecordPreview(string, SzFlag?)

Describes the features resulting from the hypothetical load of a record.

```
[SzConfigRetryable]
string GetRecordPreview(string recordDefinition, SzFlag? flags =
SzFlag.SzEntityIncludeRecordFeatureDetails)
```

Parameters

recordDefinition [string](#)

The [string](#) that defines the record, typically in JSON format.

flags [SzFlag](#)?

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzRecordFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter

will default its value to [SzRecordPreviewDefaultFlags](#). Specifying `null` is equivalent to specifying [SzNoFlags](#).

Returns

[string](#) ↗

The JSON `string` result produced by getting a record preview (depending on the specified flags).

Examples

Usage:

```
// How to get a record preview
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get a record definition (varies by application)
    string recordDefinition =
        """
        {
            "DATA_SOURCE": "TEST",
            "RECORD_ID": "DEF456",
            "NAME_FULL": "John Doe",
            "PHONE_NUMBER": "702-555-1212",
            "EMAIL_ADDRESS": "johndoe@nowhere.com"
        }
        """;
}

// get the record preview
String preview = engine.GetRecordPreview(
    recordDefinition, SzRecordPreviewDefaultFlags);

// do something with the response JSON (varies by application)
JsonObject? jsonObject = JsonNode.Parse(preview)?.AsObject();
if (jsonObject != null && jsonObject.ContainsKey("FEATURES"))
{
    JsonObject? featuresObj = jsonObject["FEATURES"]?.AsObject();
```

```

    if (featuresObj != null)
    {
        foreach (KeyValuePair<string, JsonNode?> pair in featuresObj)
        {
            string featureName = pair.Key;
            . .
        }
    }
}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to get a record preview.", e);
}

```

Example Result:

(formatted for readability)

```
{
  "FEATURES": [
    "NAME": [
      {
        "LIB_FEAT_ID": -2,
        "FEAT_DESC": "John Doe",
        "ATTRIBUTES": {
          "NAME_FULL": "John Doe"
        }
      }
    ],
    "PHONE": [
      {
        "LIB_FEAT_ID": 100002,
        "FEAT_DESC": "702-555-1212",
        "ATTRIBUTES": {
          "PHONE_NUMBER": "702-555-1212"
        }
      }
    ],
    "EMAIL": [
      {
        "LIB_FEAT_ID": -3,
        "FEAT_DESC": "johndoe@nowhere.com",
        "ATTRIBUTES": {
          "EMAIL_ADDRESS": "johndoe@nowhere.com"
        }
      }
    ]
}
```

```
        }
    }
}
}
```

Remarks

This method is used to preview the features for a record that has not been loaded.

The optionally specified bitwise-OR'd [SzFlag](#) values not only controls how the operation is performed, but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzRecordPreviewFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzException](#)

If a failure occurs.

See Also

[SzRecordPreviewDefaultFlags](#), [SzRecordPreviewFlags](#)

GetRedoRecord()

Retrieves and removes a pending redo record.

```
[SzConfigRetryable]
string GetRedoRecord()
```

Returns

[string](#) ↗

The retrieved redo record or [null](#) if there were no pending redo records.

Examples

Usage:

```

// How to check for and process redo records
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // loop through any redo records
    for (string redoRecord = engine.GetredoRecord();
        redoRecord != null;
        redoRecord = engine.GetredoRecord())
    {
        try
        {
            // process the redo record
            string info = engine.ProcessredoRecord(redoRecord, SzWithInfo);

            // do something with the "info JSON" (varies by application)
            JsonObject? jsonObject = JsonNode.Parse(info)?.AsObject();
            if (jsonObject != null && jsonObject.ContainsKey("AFFECTED_ENTITIES"))
            {
                JSONArray? affectedArr = jsonObject["AFFECTED_ENTITIES"]?.AsArray();
                for (int index = 0; affectedArr != null && index <
affectedArr.Count; index++)
                {
                    JsonObject? affected = affectedArr[index]?.AsObject();
                    long affectedID = affected?["ENTITY_ID"]?.GetValue<long>()
?? 0L;

                    . . .

                }
            }
        }
        catch (SzException e)
        {
            // handle or rethrow the other exceptions
            LogError("Failed to process redo record: " + redoRecord, e);
        }
    }

    // get the redo count
    long redoCount = engine.CountredoRecords();
}

```

```
// do something with the redo count
Log("Pending Redos: " + redoCount);
}
catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to process redos.", e);
}
```

Remarks

A `null` value will be returned if there are no pending redo records. Use [ProcessRedoRecord\(string, SzFlag?\)](#) to process the result of this method. Once a redo record is retrieved, it is no longer tracked by Senzing. The redo record may be stored externally for later processing.

This method is used in conjunction with [ProcessRedoRecord\(string, SzFlag?\)](#) and [CountredoRecords\(\)](#).

Exceptions

[SzException](#)

If a failure occurs.

See Also

[ProcessRedoRecord\(string, SzFlag?\)](#), [CountredoRecords\(\)](#),
[Code Snippet: Processing Redos while Loading](#),
[Code Snippet: Continuous Redo Processing](#),
[Code Snippet: Continuous Redo Processing via Futures](#),
[Code Snippet: Continuous Redo "With Info" Processing](#)

GetStats()

Gets and resets the internal engine workload statistics for the current operating system process.

```
string GetStats()
```

Returns

string

The `string` describing the statistics as JSON.

Examples

Usage:

```
// How to get engine stats after loading records
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // load some records to prime the stats
    . . .

    // get the stats
    string stats = engine.GetStats();

    // do something with the stats
    Log(stats);

}
catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to load records with stats.", e);
}
```

Example Result:

The example result is rather large, but can be viewed [here](#) (formatted for readability).

Remarks

The output is helpful when interacting with Senzing support. Best practice to periodically log the results.

Exceptions

[SzException](#)

If a failure occurs.

See Also

[Code Snippet: Load with Stats](#)

GetVirtualEntity(ISet<(string dataSourceCode, string recordID)>, SzFlag?)

Describes how an entity would look if composed of a given set of records.

```
[SzConfigRetryable]
string GetVirtualEntity(ISet<(string dataSourceCode, string recordID)> recordKeys,
SzFlag? flags = SzFlag.SzEntityIncludeRepresentativeFeatures |
SzFlag.SzEntityIncludeEntityName | SzFlag.SzEntityIncludeRecordSummary |
SzFlag.SzEntityIncludeRecordData | SzFlag.SzEntityIncludeRecordMatchingInfo)
```

Parameters

recordKeys [ISet](#)<(string dataSourceCode, string recordID)>

The non-null non-empty [ISet<T>](#) of tuples of data source code and record ID pairs identifying the records to use to build the virtual entity.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzVirtualEntityFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter defaults its value to [SzWhyEntitiesDefaultFlags](#) for the default recommended flags. Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) describing the virtual entity having the specified constituent records.

Examples

Usage:

```

// How to retrieve a virtual entity via a set of record keys
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get the records to operate on (varies by application)
    ISet<(string, string)> recordKeys = GetVirtualEntityRecordKeys();

    // retrieve the virtual entity for the record keys
    string result = engine.GetVirtualEntity(recordKeys,
SzVirtualEntityDefaultFlags);

    // do something with the response JSON (varies by application)
    JsonObject? jsonObject = JsonNode.Parse(result)?.AsObject();
    JsonObject? entity = jsonObject?["RESOLVED_ENTITY"]?.AsObject();
    string? entityName = entity?["ENTITY_NAME"]?.GetValue<string>();

    . . .

    if (jsonObject != null && jsonObject.ContainsKey("RECORDS"))
    {
        JSONArray? recordArr = jsonObject["RECORDS"]?.AsArray();

        for (int index = 0; recordArr != null && index < recordArr.Count; index++)
        {
            JsonObject? record = recordArr[index]?.AsObject();

            string? dataSource = record?["DATA_SOURCE"]?.GetValue<string>();
            string? recordID = record?["RECORD_ID"]?.GetValue<string>();

            . . .

        }
    }
}

catch (SzUnknownDataSourceException e)
{
    // handle the unknown data source exception
    LogError("Expected data source is not configured.", e);
}

catch (SzNotFoundException e)

```

```

{
    // handle the not-found exception
    LogError("Specified record key was not found.", e);

}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to retrieve virtual entity.", e);
}

```

Example Result: (formatted for readability)

```

{
    "RESOLVED_ENTITY": {
        "ENTITY_ID": 2,
        "ENTITY_NAME": "Joann Smith",
        "FEATURES": {
            "ADDRESS": [
                {
                    "FEAT_DESC": "101 Fifth Ave, Los Angeles, CA 90018",
                    "LIB_FEAT_ID": 22,
                    "FEAT_DESC_VALUES": [
                        {
                            "FEAT_DESC": "101 Fifth Ave, Los Angeles, CA 90018",
                            "LIB_FEAT_ID": 22
                        }
                    ]
                },
                {
                    "FEAT_DESC": "400 River Street, Pasadena, CA 90034",
                    "LIB_FEAT_ID": 65,
                    "FEAT_DESC_VALUES": [
                        {
                            "FEAT_DESC": "400 River Street, Pasadena, CA 90034",
                            "LIB_FEAT_ID": 65
                        }
                    ]
                }
            ],
            "DOB": [
                {
                    "FEAT_DESC": "15-MAR-1982",
                    "LIB_FEAT_ID": 21,
                    "FEAT_DESC_VALUES": [

```

```
        {
            "FEAT_DESC": "15-MAR-1982",
            "LIB_FEAT_ID": 21
        }
    ]
},
{
    "FEAT_DESC": "17-DEC-1977",
    "LIB_FEAT_ID": 48,
    "FEAT_DESC_VALUES": [
        {
            "FEAT_DESC": "17-DEC-1977",
            "LIB_FEAT_ID": 48
        }
    ]
},
{
    "FEAT_DESC": "23-MAY-1973",
    "LIB_FEAT_ID": 64,
    "FEAT_DESC_VALUES": [
        {
            "FEAT_DESC": "23-MAY-1973",
            "LIB_FEAT_ID": 64
        }
    ]
},
],
"NAME": [
{
    "FEAT_DESC": "Jane Donaldson",
    "LIB_FEAT_ID": 63,
    "FEAT_DESC_VALUES": [
        {
            "FEAT_DESC": "Jane Donaldson",
            "LIB_FEAT_ID": 63
        }
    ]
},
{
    "FEAT_DESC": "Joann Smith",
    "LIB_FEAT_ID": 20,
    "FEAT_DESC_VALUES": [
        {
            "FEAT_DESC": "Joann Smith",
            "LIB_FEAT_ID": 20
        }
    ]
}
]
```

```
        ],
    },
    {
        "FEAT_DESC": "John Parker",
        "LIB_FEAT_ID": 47,
        "FEAT_DESC_VALUES": [
            {
                "FEAT_DESC": "John Parker",
                "LIB_FEAT_ID": 47
            }
        ]
    }
],
"PHONE": [
    {
        "FEAT_DESC": "818-222-3131",
        "LIB_FEAT_ID": 66,
        "USAGE_TYPE": "HOME",
        "FEAT_DESC_VALUES": [
            {
                "FEAT_DESC": "818-222-3131",
                "LIB_FEAT_ID": 66
            }
        ]
    },
    {
        "FEAT_DESC": "818-888-3939",
        "LIB_FEAT_ID": 23,
        "USAGE_TYPE": "HOME",
        "FEAT_DESC_VALUES": [
            {
                "FEAT_DESC": "818-888-3939",
                "LIB_FEAT_ID": 23
            }
        ]
    },
    {
        "FEAT_DESC": "818-999-2121",
        "LIB_FEAT_ID": 49,
        "USAGE_TYPE": "HOME",
        "FEAT_DESC_VALUES": [
            {
                "FEAT_DESC": "818-999-2121",
                "LIB_FEAT_ID": 49
            }
        ]
    }
]
```

```
        },
        {
          "FEAT_DESC": "213-555-1212",
          "LIB_FEAT_ID": 5,
          "USAGE_TYPE": "MOBILE",
          "FEAT_DESC_VALUES": [
            {
              "FEAT_DESC": "213-555-1212",
              "LIB_FEAT_ID": 5
            }
          ]
        },
        {
          "FEAT_DESC": "818-555-1313",
          "LIB_FEAT_ID": 50,
          "USAGE_TYPE": "MOBILE",
          "FEAT_DESC_VALUES": [
            {
              "FEAT_DESC": "818-555-1313",
              "LIB_FEAT_ID": 50
            }
          ]
        }
      ],
      "RECORD_SUMMARY": [
        {
          "DATA_SOURCE": "PASSENGERS",
          "RECORD_COUNT": 3
        }
      ],
      "RECORDS": [
        {
          "DATA_SOURCE": "PASSENGERS",
          "RECORD_ID": "DEF456",
          "INTERNAL_ID": 2
        },
        {
          "DATA_SOURCE": "PASSENGERS",
          "RECORD_ID": "GHI789",
          "INTERNAL_ID": 3
        },
        {
          "DATA_SOURCE": "PASSENGERS",
          "RECORD_ID": "JKL012",
          "INTERNAL_ID": 4
        }
      ]
    }
  ]
}
```

```
        }
    ]
}
}
```

Remarks

Virtual entities do not have relationships.

The optionally specified bitwise-OR'd [SzFlag](#) values not only control how the operation is performed but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzVirtualEntityFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzUnknownDataSourceException](#)

If an unrecognized data source code is specified.

[SzNotFoundException](#)

If any of the records for the specified data source code and record ID pairs cannot be found.

[SzException](#)

If a failure occurs.

See Also

[SzVirtualEntityDefaultFlags](#), [SzHowFlags](#), [HowEntity\(long, SzFlag?\)](#)

HowEntity(long, SzFlag?)

Explains how an entity was constructed from its records.

```
[SzConfigRetryable]
string HowEntity(long entityID, SzFlag? flags = SzFlag.SzIncludeFeatureScores)
```

Parameters

entityID [long](#)

The entity ID of the entity.

flags [SzFlag](#)?

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzHowFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter defaults its value to [SzHowEntityDefaultFlags](#) for the default recommended flags. Specifying `null` is equivalent to specifying [SzNoFlags](#).

Returns

[string](#) ↗

The JSON [string](#) describing the how the entity was constructed.

Examples

Usage:

```
// How to retrieve the "how analysis" for an entity via its entity ID
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get the entity ID on which to operate (varies by application)
    long entityID = GetEntityID();

    // determine how the entity was formed
    string results = engine.HowEntity(entityID, SzHowEntityDefaultFlags);

    // do something with the response JSON (varies by application)
    JsonObject? jsonObject = JsonNode.Parse(results)?.AsObject();
    JsonObject? howResults = jsonObject?["HOW_RESULTS"]?.AsObject();
    JsonArray? stepsArr = howResults?["RESOLUTION_STEPS"]?.AsArray();

    for (int index = 0; stepsArr != null && index < stepsArr.Count; index++)
    {
        JsonObject? step = stepsArr[index]?.AsObject();

        JsonObject? matchInfo = step?["MATCH_INFO"]?.AsObject();
```

```

    . . .

}

} catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Entity not found for entity ID.", e);

}

} catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to retrieve how analysis.", e);
}

```

Example Result: (formatted for readability)

```
{
    "HOW_RESULTS": {
        "RESOLUTION_STEPS": [
            {
                "STEP": 1,
                "VIRTUAL_ENTITY_1": {
                    "VIRTUAL_ENTITY_ID": "V100002",
                    "MEMBER_RECORDS": [
                        {
                            "INTERNAL_ID": 100002,
                            "RECORDS": [
                                {
                                    "DATA_SOURCE": "TEST",
                                    "RECORD_ID": "ABC123"
                                }
                            ]
                        }
                    ]
                },
                "VIRTUAL_ENTITY_2": {
                    "VIRTUAL_ENTITY_ID": "V100003",
                    "MEMBER_RECORDS": [
                        {
                            "INTERNAL_ID": 100003,
                            "RECORDS": [
                                {
                                    "DATA_SOURCE": "TEST",

```

```

        "RECORD_ID": "XYZ987"
    }
]
}
],
},
"INBOUND_VIRTUAL_ENTITY_ID": "V100003",
"RESULT_VIRTUAL_ENTITY_ID": "V100002-S1",
"MATCH_INFO": {
    "MATCH_KEY": "\u002BNAME\u002BEMAIL",
    "ERRULE_CODE": "SF1_CNAME",
    "CANDIDATE_KEYS": {
        "EMAIL_KEY": [
            {
                "FEAT_ID": 100007,
                "FEAT_DESC": "joeschmoe@NOWHERE.COM"
            }
        ],
        "NAME_KEY": [
            {
                "FEAT_ID": 100005,
                "FEAT_DESC": "JSF|XM"
            }
        ]
    },
    "FEATURE_SCORES": {
        "EMAIL": [
            {
                "INBOUND_FEAT_ID": 100003,
                "INBOUND_FEAT_DESC": "joeschmoe@nowhere.com",
                "CANDIDATE_FEAT_ID": 100003,
                "CANDIDATE_FEAT_DESC": "joeschmoe@nowhere.com",
                "SCORE": 100,
                "ADDITIONAL_SCORES": {
                    "FULL_SCORE": 100
                },
                "SCORE_BUCKET": "SAME",
                "SCORE_BEHAVIOR": "F1"
            }
        ],
        "NAME": [
            {
                "INBOUND_FEAT_ID": 100010,
                "INBOUND_FEAT_DESC": "Joseph Schmoe",
                "CANDIDATE_FEAT_ID": 100001,
                "CANDIDATE_FEAT_DESC": "Joe Schmoe",
                "SCORE": 100
            }
        ]
    }
}
]
```

```
"SCORE": 98,
"ADDITIONAL_SCORES": {
    "GNR_FN": 98,
    "GNR_SN": -1,
    "GNR_GN": -1,
    "GENERATION_MATCH": -1,
    "GNR_ON": -1
},
"SCORE_BUCKET": "CLOSE",
"SCORE_BEHAVIOR": "NAME"
}
],
"PHONE": [
{
    "INBOUND_FEAT_ID": 100012,
    "INBOUND_FEAT_DESC": "702-555-1313",
    "INBOUND_FEAT_USAGE_TYPE": "WORK",
    "CANDIDATE_FEAT_ID": 100002,
    "CANDIDATE_FEAT_DESC": "702-555-1212",
    "SCORE": 85,
    "ADDITIONAL_SCORES": {
        "FULL_SCORE": 85
    },
    "SCORE_BUCKET": "LIKELY",
    "SCORE_BEHAVIOR": "FF"
}
]
}
},
{
    "STEP": 2,
    "VIRTUAL_ENTITY_1": {
        "VIRTUAL_ENTITY_ID": "V100002-S1",
        "MEMBER_RECORDS": [
{
        "INTERNAL_ID": 100002,
        "RECORDS": [
{
            "DATA_SOURCE": "TEST",
            "RECORD_ID": "ABC123"
}
]
},
{
        "INTERNAL_ID": 100003,

```

```

    "RECORDS": [
        {
            "DATA_SOURCE": "TEST",
            "RECORD_ID": "XYZ987"
        }
    ]
},
"VIRTUAL_ENTITY_2": {
    "VIRTUAL_ENTITY_ID": "V100004",
    "MEMBER_RECORDS": [
        {
            "INTERNAL_ID": 100004,
            "RECORDS": [
                {
                    "DATA_SOURCE": "TEST",
                    "RECORD_ID": "ZYX789"
                }
            ]
        }
    ]
},
"INBOUND_VIRTUAL_ENTITY_ID": "V100002-S1",
"RESULT_VIRTUAL_ENTITY_ID": "V100002-S2",
"MATCH_INFO": {
    "MATCH_KEY": "\u002BNAME\u002BADDRESS\u002BPHONE",
    "ERRULE_CODE": "MFF_CNAME",
    "CANDIDATE_KEYS": {
        "ADDR_KEY": [
            {
                "FEAT_ID": 100013,
                "FEAT_DESC": "101|MN| |89101"
            },
            {
                "FEAT_ID": 100014,
                "FEAT_DESC": "101|MN| |LS FKS"
            }
        ],
        "NAMEADDR_KEY": [
            {
                "FEAT_ID": 100016,
                "FEAT_DESC": "JSF|XM|ADDR_KEY.EXPRESSION=101|MN| |LS FKS"
            },
            {
                "FEAT_ID": 100017,

```

```

        "FEAT_DESC": "JSF|XM|ADDR_KEY.EXPRESSION=101|MN|89101"
    }
],
"NAMEPHONE_KEY": [
{
    "FEAT_ID": 100008,
    "FEAT_DESC": "JSF|XM|PHONE.PHONE_LAST_5=51212"
},
{
    "FEAT_ID": 100020,
    "FEAT_DESC": "JSF|XM|PHONE.PHONE_LAST_5=51313"
}
],
"NAMEREGION_KEY": [
{
    "FEAT_ID": 100018,
    "FEAT_DESC": "JSF|XM|POST=89101"
},
{
    "FEAT_ID": 100019,
    "FEAT_DESC": "JSF|XM|ADDRESS.CITY_STD=LS FKS"
}
],
"NAME_KEY": [
{
    "FEAT_ID": 100005,
    "FEAT_DESC": "JSF|XM"
}
],
"PHONE_KEY": [
{
    "FEAT_ID": 100006,
    "FEAT_DESC": "7025551212"
},
{
    "FEAT_ID": 100015,
    "FEAT_DESC": "7025551313"
}
]
},
"FEATURE_SCORES": {
"ADDRESS": [
{
    "INBOUND_FEAT_ID": 100011,
    "INBOUND_FEAT_DESC": "101 Main St.; Las Vegas, NV 89101",
    "CANDIDATE_FEAT_ID": 100011,

```

```

    "CANDIDATE_FEAT_DESC": "101 Main St.; Las Vegas, NV 89101",
    "SCORE": 100,
    "ADDITIONAL_SCORES": {
        "FULL_SCORE": 100
    },
    "SCORE_BUCKET": "SAME",
    "SCORE_BEHAVIOR": "FF"
}
],
"NAME": [
{
    "INBOUND_FEAT_ID": 100010,
    "INBOUND_FEAT_DESC": "Joseph Schmoe",
    "CANDIDATE_FEAT_ID": 100021,
    "CANDIDATE_FEAT_DESC": "Joseph W Schmoe",
    "SCORE": 92,
    "ADDITIONAL_SCORES": {
        "GNR_FN": 92,
        "GNR_SN": -1,
        "GNR_GN": -1,
        "GENERATION_MATCH": -1,
        "GNR_ON": -1
    },
    "SCORE_BUCKET": "CLOSE",
    "SCORE_BEHAVIOR": "NAME"
}
],
"PHONE": [
{
    "INBOUND_FEAT_ID": 100002,
    "INBOUND_FEAT_DESC": "702-555-1212",
    "CANDIDATE_FEAT_ID": 100002,
    "CANDIDATE_FEAT_DESC": "702-555-1212",
    "CANDIDATE_FEAT_USAGE_TYPE": "HOME",
    "SCORE": 100,
    "ADDITIONAL_SCORES": {
        "FULL_SCORE": 100
    },
    "SCORE_BUCKET": "SAME",
    "SCORE_BEHAVIOR": "FF"
}
]
}
],
}
],

```

```

"FINAL_STATE": {
  "NEED_REEVALUATION": 0,
  "VIRTUAL_ENTITIES": [
    {
      "VIRTUAL_ENTITY_ID": "V100002-S2",
      "MEMBER_RECORDS": [
        {
          "INTERNAL_ID": 100002,
          "RECORDS": [
            {
              "DATA_SOURCE": "TEST",
              "RECORD_ID": "ABC123"
            }
          ]
        },
        {
          "INTERNAL_ID": 100003,
          "RECORDS": [
            {
              "DATA_SOURCE": "TEST",
              "RECORD_ID": "XYZ987"
            }
          ]
        },
        {
          "INTERNAL_ID": 100004,
          "RECORDS": [
            {
              "DATA_SOURCE": "TEST",
              "RECORD_ID": "ZYX789"
            }
          ]
        }
      ]
    }
  }
}

```

Remarks

The optionally specified bitwise-OR'd [SzFlag](#) values not only control how the operation is performed but also the content of the response. Any [SzFlag](#) value may be included, but

only flags belonging to the [SzHowFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzNotFoundException](#)

If the entity for the specified entity ID could not be found.

[SzException](#)

If a failure occurs.

See Also

[SzHowEntityDefaultFlags](#), [SzHowFlags](#),
[GetVirtualEntity\(ISet<\(string dataSourceCode, string recordID\)>, SzFlag?\)](#).

PrimeEngine()

Pre-loads engine resources.

```
void PrimeEngine()
```

Examples

Usage:

```
// How to prime the SzEngine to expedite subsequent operations
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // prime the engine
    engine.PrimeEngine();

    // use the primed engine to perform additional tasks
    . . .

}
```

```
catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to prime engine.", e);
}
```

Remarks

Explicitly calling this method ensures the performance cost is incurred at a predictable time rather than unexpectedly with the first call requiring the resources.

Exceptions

[SzException](#)

If a failure occurs.

See Also

[Code Snippet: Engine Priming](#)

ProcessRedoRecord(string, SzFlag?)

Processes the provided redo record.

```
[SzConfigRetryable]
string ProcessRedoRecord(string redoRecord, SzFlag? flags = (SzFlag)0)
```

Parameters

[redoRecord string](#)

The redo record to be processed.

[flags SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzRedoFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter will default its value to [SzRedoDefaultFlags](#). Specify [SzWithInfo](#) for an INFO response. Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

string

The JSON `string` result produced by processing the redo record, or `null` if the specified flags do not indicate that an INFO message should be returned.

Examples

Usage:

```
// How to check for and process redo records
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // loop through any redo records
    for (string redoRecord = engine.GetredoRecord();
        redoRecord != null;
        redoRecord = engine.GetredoRecord())
    {
        try
        {
            // process the redo record
            string info = engine.ProcessredoRecord(redoRecord, SzWithInfo);

            // do something with the "info JSON" (varies by application)
            JsonObject? jsonObject = JsonNode.Parse(info)?.AsObject();
            if (jsonObject != null && jsonObject.ContainsKey("AFFECTED_ENTITIES"))
            {
                JSONArray? affectedArr = jsonObject["AFFECTED_ENTITIES"]?.AsArray();
                for (int index = 0; affectedArr != null && index <
affectedArr.Count; index++)
                {
                    JsonObject? affected = affectedArr[index]?.AsObject();
                    long affectedID = affected?["ENTITY_ID"]?.GetValue<long>()
?? 0L;

                    . . .

                }
            }
        }
        catch (SzException e)
```

```

    {
        // handle or rethrow the other exceptions
        LogError("Failed to process redo record: " + redoRecord, e);
    }
}

// get the redo count
long redoCount = engine.CountRedoRecords();

// do something with the redo count
Log("Pending Redos: " + redoCount);
}

catch (SzException e)
{
    // handle or rethrow the other exceptions
    LogError("Failed to process redos.", e);
}

```

Example Info Result: (formatted for readability)

```
{
  "DATA_SOURCE": "TEST",
  "RECORD_ID": "ABC123",
  "AFFECTED_ENTITIES": [
    {
      "ENTITY_ID": 100002
    }
  ]
}
```

Remarks

This operation performs entity resolution. Calling this method has the potential to create more redo records in certain situations.

This method is used in conjunction with [GetRedoRecord\(\)](#) and [CountRedoRecords\(\)](#).

The optionally specified bitwise-OR'd [SzFlag](#) values not only controls how the operation is performed, but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzRedoFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzException](#)

If a failure occurs.

See Also

[SzWithInfo](#), [SzRedoFlags](#), [SzredoDefaultFlags](#), [GetRedoRecord\(\)](#), [CountRedoRecords\(\)](#),
[Code Snippet: Processing Redos while Loading](#),
[Code Snippet: Continuous Redo Processing](#),
[Code Snippet: Continuous Redo Processing via Futures](#),
[Code Snippet: Continuous Redo "With Info" Processing](#)

ReevaluateEntity(long, SzFlag?)

Reevaluates an entity by entity ID.

```
[SzConfigRetryable]  
string ReevaluateEntity(long entityID, SzFlag? flags = (SzFlag)0)
```

Parameters

entityID [long](#)

The ID of the resolved entity to reevaluate.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzReevaluateEntityFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter will default its value to [SzReevaluateEntityDefaultFlags](#). Specify [SzWithInfo](#) for an INFO response. Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) result produced by reevaluating the entity, or [null](#) if the specified flags do not indicate that an INFO message should be returned.

Examples

Usage:

```

// How to reevaluate an entity
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get the ID of an entity to reevaluate (varies by application)
    long entityID = GetEntityID();

    // reevaluate an entity in the repository
    string info = engine.ReevaluateEntity(entityID, SzWithInfo);

    // do something with the "info JSON" (varies by application)
    JsonObject? jsonObject = JsonNode.Parse(info)?.AsObject();
    if (jsonObject != null && jsonObject.ContainsKey("AFFECTED_ENTITIES"))
    {
        JSONArray? affectedArr = jsonObject["AFFECTED_ENTITIES"]?.AsArray();

        for (int index = 0; affectedArr != null && index <
affectedArr.Count; index++)
        {
            JsonObject? affected = affectedArr[index]?.AsObject();

            long affectedID = affected?["ENTITY_ID"]?.GetValue<long>() ?? 0L;

            . . .
        }
    }
}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to reevaluate entity.", e);
}

```

Example Result: (formatted for readability)

```
{
  "AFFECTED_ENTITIES": [
    {
      "ENTITY_ID": 100002
    }
  ]
}
```

```
    }  
]  
}
```

Remarks

This operation performs entity resolution. If the entity is not found, then no changes are made.

The optionally specified bitwise-OR'd [SzFlag](#) values not only controls how the operation is performed, but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzReevaluateEntityFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzException](#)

If a failure occurs.

See Also

[SzNoFlags](#), [SzWithInfo](#), [SzReevaluateEntityDefaultFlags](#), [SzReevaluateEntityFlags](#)

ReevaluateRecord(string, string, SzFlag?)

Reevaluates an entity by record ID.

```
[SzConfigRetryable]  
string ReevaluateRecord(string dataSourceCode, string recordID, SzFlag? flags  
= (SzFlag)0)
```

Parameters

dataSourceCode [string](#)

The data source code identifying the data source for the record to reevaluate.

recordID [string](#)

The record ID that uniquely identifies the record to reevaluate within the scope of its associated data source.

flags [SzFlag](#)?

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzReevaluateRecordFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter will default its value to [SzReevaluateRecordDefaultFlags](#). Specify [SzWithInfo](#) for an INFO response. Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[string](#) ↗

The JSON [string](#) result produced by reevaluating the record, or [null](#) if the specified flags do not indicate that an INFO message should be returned.

Examples

Usage:

```
// How to reevaluate a record
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // reevaluate a record in the repository
    string info = engine.ReevaluateRecord("TEST", "ABC123", SzWithInfo);

    // do something with the "info JSON" (varies by application)
    JsonObject? jsonObject = JsonNode.Parse(info)?.AsObject();
    if (jsonObject != null && jsonObject.ContainsKey("AFFECTED_ENTITIES"))
    {
        JSONArray? affectedArr = jsonObject["AFFECTED_ENTITIES"]?.AsArray();

        for (int index = 0; affectedArr != null && index <
affectedArr.Count; index++)
        {
            JsonObject? affected = affectedArr[index]?.AsObject();

            long affectedID = affected?["ENTITY_ID"]?.GetValue<long>() ?? 0L;

            . . .
        }
    }
}
```

```

    }

}

catch (SzUnknownDataSourceException e)
{
    // handle the unknown data source exception
    LogError("Expected data source is not configured.", e);

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to reevaluate record.", e);
}

```

Example Result:

(formatted for readability)

```
{
  "DATA_SOURCE": "TEST",
  "RECORD_ID": "ABC123",
  "AFFECTED_ENTITIES": [
    {
      "ENTITY_ID": 100002
    }
  ]
}
```

Remarks

This operation performs entity resolution. If the record is not found, then no changes are made.

The optionally specified bitwise-OR'd [SzFlag](#) values not only controls how the operation is performed, but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzReevaluateRecordFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzUnknownDataSourceException](#)

If an unrecognized data source code is specified.

[SzException](#)

If a failure occurs.

See Also

[SzNoFlags](#), [SzWithInfo](#), [SzReevaluateRecordDefaultFlags](#), [SzReevaluateRecordFlags](#)

SearchByAttributes(string, SzFlag?)

Convenience method for calling [SearchByAttributes\(string, string, SzFlag?\)](#) with a `null` value for the search profile parameter.

```
[SzConfigRetryable]
string SearchByAttributes(string attributes, SzFlag? flags =
SzFlag.SzExportIncludeMultiRecordEntities | SzFlag.SzExportIncludePossiblySame |
SzFlag.SzExportIncludePossiblyRelated | SzFlag.SzExportIncludeNameOnly |
SzFlag.SzEntityIncludeRepresentativeFeatures | SzFlag.SzEntityIncludeEntityName
| SzFlag.SzEntityIncludeRecordSummary | SzFlag.SzIncludeFeatureScores
| SzFlag.SzSearchIncludeStats)
```

Parameters

attributes [string](#)

The search attributes defining the hypothetical record to match and/or relate to in order to obtain the search results.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzSearchFlags](#) group to control how the operation is performed and the content of the response, omitting this parameter will default its value to [SzSearchByAttributesDefaultFlags](#) for the default recommended flags. Specifying `null` is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The resulting JSON `string` describing the result of the search.

Examples

Usage:

```

// How to search for entities matching criteria
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get the search attributes (varies by application)
    string searchAttributes =
        """
        {
            "NAME_FULL": "Joe Schmoe",
            "PHONE_NUMBER": "702-555-1212",
            "EMAIL_ADDRESS": "joeschmoe@nowhere.com"
        }
        """;

    // search for matching entities in the repository
    string results = engine.SearchByAttributes(
        searchAttributes,
        SzSearchByAttributesDefaultFlags);

    // do something with the response JSON (varies by application)
    JsonObject? jsonObject = JsonNode.Parse(results)?.AsObject();
    if (jsonObject != null && jsonObject.ContainsKey("RESOLVED_ENTITIES"))
    {
        JSONArray? resultsArr = jsonObject["RESOLVED_ENTITIES"]?.AsArray();

        for (int index = 0; resultsArr != null && index < resultsArr.Count; index++)
        {
            JsonObject? result = resultsArr[index]?.AsObject();

            // get the match info for the result
            JsonObject? matchInfo = result?["MATCH_INFO"]?.AsObject();

            . .

            // get the entity for the result
            JsonObject? entityInfo = result?["ENTITY"]?.AsObject();
            JsonObject? entity = entityInfo?["RESOLVED_ENTITY"]?.AsObject();
            long entityID = entity?["ENTITY_ID"]?.GetValue<long>() ?? 0L;

            . .
        }
    }
}

```

```

    }

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to search for entities.", e);
}

```

Example Result: (formatted for readability)

```
{
  "RESOLVED_ENTITIES": [
    {
      "MATCH_INFO": {
        "MATCH_LEVEL_CODE": "RESOLVED",
        "MATCH_KEY": "\u002BNAME\u002BPHONE\u002BEMAIL",
        "ERRULE_CODE": "SF1_PNAME_CFF",
        "CANDIDATE_KEYS": {
          "EMAIL_KEY": [
            {
              "FEAT_ID": 100007,
              "FEAT_DESC": "joeschmoe@NOWHERE.COM"
            }
          ],
          "NAMEPHONE_KEY": [
            {
              "FEAT_ID": 100008,
              "FEAT_DESC": "JSF|XM|PHONE.PHONE_LAST_5=51212"
            },
            {
              "FEAT_ID": 100009,
              "FEAT_DESC": "JOE|XM|PHONE.PHONE_LAST_5=51212"
            }
          ],
          "NAME_KEY": [
            {
              "FEAT_ID": 100004,
              "FEAT_DESC": "JOE|XM"
            },
            {
              "FEAT_ID": 100005,
              "FEAT_DESC": "JSF|XM"
            }
          ],
        }
      }
    }
  ]
}
```

```

"PHONE_KEY": [
  {
    "FEAT_ID": 100006,
    "FEAT_DESC": "7025551212"
  }
],
},
"FEATURE_SCORES": {
  "EMAIL": [
    {
      "INBOUND_FEAT_ID": 100003,
      "INBOUND_FEAT_DESC": "joeschmoe@nowhere.com",
      "CANDIDATE_FEAT_ID": 100003,
      "CANDIDATE_FEAT_DESC": "joeschmoe@nowhere.com",
      "SCORE": 100,
      "ADDITIONAL_SCORES": {
        "FULL_SCORE": 100
      },
      "SCORE_BUCKET": "SAME",
      "SCORE_BEHAVIOR": "F1"
    }
  ],
  "NAME": [
    {
      "INBOUND_FEAT_ID": 100001,
      "INBOUND_FEAT_DESC": "Joe Schmoe",
      "CANDIDATE_FEAT_ID": 100001,
      "CANDIDATE_FEAT_DESC": "Joe Schmoe",
      "SCORE": 100,
      "ADDITIONAL_SCORES": {
        "GNR_FN": 100,
        "GNR_SN": -1,
        "GNR_GN": -1,
        "GENERATION_MATCH": -1,
        "GNR_ON": -1
      },
      "SCORE_BUCKET": "SAME",
      "SCORE_BEHAVIOR": "NAME"
    }
  ],
  "PHONE": [
    {
      "INBOUND_FEAT_ID": 100002,
      "INBOUND_FEAT_DESC": "702-555-1212",
      "CANDIDATE_FEAT_ID": 100002,
      "CANDIDATE_FEAT_DESC": "702-555-1212",
      "SCORE": 100
    }
  ]
}

```

```
        "SCORE": 100,
        "ADDITIONAL_SCORES": {
            "FULL_SCORE": 100
        },
        "SCORE_BUCKET": "SAME",
        "SCORE_BEHAVIOR": "FF"
    }
]
}
},
"ENTITY": {
    "RESOLVED_ENTITY": {
        "ENTITY_ID": 100002,
        "ENTITY_NAME": "Joseph W Schmoe",
        "FEATURES": {
            "ADDRESS": [
                {
                    "FEAT_DESC": "101 Main St.; Las Vegas, NV 89101",
                    "LIB_FEAT_ID": 100011,
                    "FEAT_DESC_VALUES": [
                        {
                            "FEAT_DESC": "101 Main St.; Las Vegas, NV 89101",
                            "LIB_FEAT_ID": 100011
                        }
                    ]
                }
            ],
            "EMAIL": [
                {
                    "FEAT_DESC": "joeschmoe@nowhere.com",
                    "LIB_FEAT_ID": 100003,
                    "FEAT_DESC_VALUES": [
                        {
                            "FEAT_DESC": "joeschmoe@nowhere.com",
                            "LIB_FEAT_ID": 100003
                        }
                    ]
                }
            ],
            "NAME": [
                {
                    "FEAT_DESC": "Joseph W Schmoe",
                    "LIB_FEAT_ID": 100021,
                    "FEAT_DESC_VALUES": [
                        {
                            "FEAT_DESC": "Joseph W Schmoe",
                            "LIB_FEAT_ID": 100021
                        }
                    ]
                }
            ]
        }
    }
}
```

```
        "LIB_FEAT_ID": 100021
    },
    {
        "FEAT_DESC": "Joseph Schmoe",
        "LIB_FEAT_ID": 100010
    },
    {
        "FEAT_DESC": "Joe Schmoe",
        "LIB_FEAT_ID": 100001
    }
]
}
],
"PHONE": [
{
    "FEAT_DESC": "702-555-1212",
    "LIB_FEAT_ID": 100002,
    "FEAT_DESC_VALUES": [
        {
            "FEAT_DESC": "702-555-1212",
            "LIB_FEAT_ID": 100002
        }
    ]
},
{
    "FEAT_DESC": "702-555-1212",
    "LIB_FEAT_ID": 100002,
    "USAGE_TYPE": "HOME",
    "FEAT_DESC_VALUES": [
        {
            "FEAT_DESC": "702-555-1212",
            "LIB_FEAT_ID": 100002
        }
    ]
},
{
    "FEAT_DESC": "702-555-1313",
    "LIB_FEAT_ID": 100012,
    "USAGE_TYPE": "WORK",
    "FEAT_DESC_VALUES": [
        {
            "FEAT_DESC": "702-555-1313",
            "LIB_FEAT_ID": 100012
        }
    ]
}
]
```

```
        ],
    },
    "RECORD_SUMMARY": [
        {
            "DATA_SOURCE": "TEST",
            "RECORD_COUNT": 3
        }
    ]
}
],
"SEARCH_STATISTICS": [
{
    "CANDIDATE_KEYS": {
        "FEATURE_TYPES": [
            {
                "FTYPE_CODE": "NAME_KEY",
                "FOUND": 2,
                "NOT_FOUND": 0,
                "GENERIC": 0
            },
            {
                "FTYPE_CODE": "PHONE_KEY",
                "FOUND": 1,
                "NOT_FOUND": 0,
                "GENERIC": 0
            },
            {
                "FTYPE_CODE": "EMAIL_KEY",
                "FOUND": 1,
                "NOT_FOUND": 0,
                "GENERIC": 0
            },
            {
                "FTYPE_CODE": "NAMEPHONE_KEY",
                "FOUND": 2,
                "NOT_FOUND": 0,
                "GENERIC": 0
            }
        ],
        "SUMMARY": {
            "FOUND": 6,
            "NOT_FOUND": 0,
            "GENERIC": 0
        }
    }
}
```

```
        }
    }
]
```

Remarks

See [SearchByAttributes\(string, string, SzFlag?\)](#) documentation for details.

Exceptions

[SzException](#)

If a failure occurs.

See Also

[SearchByAttributes\(string, string, SzFlag?\)](#), [SzSearchByAttributesDefaultFlags](#),
[SzSearchByAttributesAll](#), [SzSearchByAttributesStrong](#), [SzSearchByAttributesMinimalAll](#),
[SzSearchByAttributesMinimalStrong](#), [SzSearchFlags](#), [Code Snippet: Search Records](#),
[Code Snippet: Search via Futures](#)

SearchByAttributes(string, string, SzFlag?)

Searches for entities that match or relate to the provided attributes.

```
[SzConfigRetryable]
string SearchByAttributes(string attributes, string searchProfile, SzFlag? flags =
SzFlag.SzExportIncludeMultiRecordEntities | SzFlag.SzExportIncludePossiblySame |
SzFlag.SzExportIncludePossiblyRelated | SzFlag.SzExportIncludeNameOnly |
SzFlag.SzEntityIncludeRepresentativeFeatures | SzFlag.SzEntityIncludeEntityName
| SzFlag.SzEntityIncludeRecordSummary | SzFlag.SzIncludeFeatureScores
| SzFlag.SzSearchIncludeStats)
```

Parameters

attributes [string](#)

The search attributes defining the hypothetical record to match and/or relate to in order to obtain the search results.

searchProfile [string](#)

The optional search profile identifier, or `null` if the default search profile should be used for the search.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzSearchFlags](#) group to control how the operation is performed and the content of the response, omitting this parameter will default its value to [SzSearchByAttributesDefaultFlags](#) for the default recommended flags. Specifying `null` is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The resulting JSON [string](#) describing the result of the search.

Examples

Usage:

```
// How to search for entities matching criteria using a search profile
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get the search attributes (varies by application)
    string searchAttributes =
        """
        {
            "NAME_FULL": "Joe Schmoe",
            "PHONE_NUMBER": "702-555-1212",
            "EMAIL_ADDRESS": "joeschmoe@nowhere.com"
        }
        """;
}

// get a search profile (varies by application)
string searchProfile = GetSearchProfile();

// search for matching entities in the repository
string results = engine.SearchByAttributes(
    searchAttributes,
```

```

    searchProfile,
    SzSearchByAttributesDefaultFlags);

// do something with the response JSON (varies by application)
JsonObject? jsonObject = JsonNode.Parse(results)?.AsObject();
if (jsonObject != null && jsonObject.ContainsKey("RESOLVED_ENTITIES"))
{
    JSONArray? resultsArr = jsonObject["RESOLVED_ENTITIES"]?.AsArray();

    for (int index = 0; resultsArr != null && index < resultsArr.Count; index++)
    {
        JsonObject? result = resultsArr[index]?.AsObject();

        // get the match info for the result
        JsonObject? matchInfo = result?["MATCH_INFO"]?.AsObject();

        . . .

        // get the entity for the result
        JsonObject? entityInfo = result?["ENTITY"]?.AsObject();
        JsonObject? entity = entityInfo?["RESOLVED_ENTITY"]?.AsObject();
        long entityID = entity?["ENTITY_ID"]?.GetValue<long>() ?? 0L;

        . . .

    }
}

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to search for entities.", e);
}

```

Example Result:

(formatted for readability)

```
{
  "RESOLVED_ENTITIES": [
    {
      "MATCH_INFO": {
        "MATCH_LEVEL_CODE": "RESOLVED",
        "MATCH_KEY": "\u002BNAME\u002BPHONE\u002BEMAIL",
        "ERRULE_CODE": "SF1_PNAME_CFF",
        "CANDIDATE_KEYS": {
          "EMAIL_KEY": [
            "test@example.com"
          ]
        }
      }
    }
  ]
}
```

```

        {
          "FEAT_ID": 100007,
          "FEAT_DESC": "joeschmoe@NOWHERE.COM"
        },
      ],
      "NAMEPHONE_KEY": [
        {
          "FEAT_ID": 100008,
          "FEAT_DESC": "JSF|XM|PHONE.PHONE_LAST_5=51212"
        },
        {
          "FEAT_ID": 100009,
          "FEAT_DESC": "JOE|XM|PHONE.PHONE_LAST_5=51212"
        }
      ],
      "NAME_KEY": [
        {
          "FEAT_ID": 100004,
          "FEAT_DESC": "JOE|XM"
        },
        {
          "FEAT_ID": 100005,
          "FEAT_DESC": "JSF|XM"
        }
      ],
      "PHONE_KEY": [
        {
          "FEAT_ID": 100006,
          "FEAT_DESC": "7025551212"
        }
      ]
    },
    "FEATURE_SCORES": {
      "EMAIL": [
        {
          "INBOUND_FEAT_ID": 100003,
          "INBOUND_FEAT_DESC": "joeschmoe@nowhere.com",
          "CANDIDATE_FEAT_ID": 100003,
          "CANDIDATE_FEAT_DESC": "joeschmoe@nowhere.com",
          "SCORE": 100,
          "ADDITIONAL_SCORES": {
            "FULL_SCORE": 100
          },
          "SCORE_BUCKET": "SAME",
          "SCORE_BEHAVIOR": "F1"
        }
      ]
    }
  }
}

```

```

],
"NAME": [
{
  "INBOUND_FEAT_ID": 100001,
  "INBOUND_FEAT_DESC": "Joe Schmoe",
  "CANDIDATE_FEAT_ID": 100001,
  "CANDIDATE_FEAT_DESC": "Joe Schmoe",
  "SCORE": 100,
  "ADDITIONAL_SCORES": {
    "GNR_FN": 100,
    "GNR_SN": -1,
    "GNR_GN": -1,
    "GENERATION_MATCH": -1,
    "GNR_ON": -1
  },
  "SCORE_BUCKET": "SAME",
  "SCORE_BEHAVIOR": "NAME"
}
],
"PHONE": [
{
  "INBOUND_FEAT_ID": 100002,
  "INBOUND_FEAT_DESC": "702-555-1212",
  "CANDIDATE_FEAT_ID": 100002,
  "CANDIDATE_FEAT_DESC": "702-555-1212",
  "SCORE": 100,
  "ADDITIONAL_SCORES": {
    "FULL_SCORE": 100
  },
  "SCORE_BUCKET": "SAME",
  "SCORE_BEHAVIOR": "FF"
}
]
},
"ENTITY": {
  "RESOLVED_ENTITY": {
    "ENTITY_ID": 100002,
    "ENTITY_NAME": "Joseph W Schmoe",
    "FEATURES": {
      "ADDRESS": [
        {
          "FEAT_DESC": "101 Main St.; Las Vegas, NV 89101",
          "LIB_FEAT_ID": 100011,
          "FEAT_DESC_VALUES": [
            {

```

```
        "FEAT_DESC": "101 Main St.; Las Vegas, NV 89101",
        "LIB_FEAT_ID": 100011
    }
]
}
],
"EMAIL": [
{
    "FEAT_DESC": "joeschmoe@nowhere.com",
    "LIB_FEAT_ID": 100003,
    "FEAT_DESC_VALUES": [
        {
            "FEAT_DESC": "joeschmoe@nowhere.com",
            "LIB_FEAT_ID": 100003
        }
    ]
},
"NAME": [
{
    "FEAT_DESC": "Joseph W Schmoe",
    "LIB_FEAT_ID": 100021,
    "FEAT_DESC_VALUES": [
        {
            "FEAT_DESC": "Joseph W Schmoe",
            "LIB_FEAT_ID": 100021
        },
        {
            "FEAT_DESC": "Joseph Schmoe",
            "LIB_FEAT_ID": 100010
        },
        {
            "FEAT_DESC": "Joe Schmoe",
            "LIB_FEAT_ID": 100001
        }
    ]
},
"PHONE": [
{
    "FEAT_DESC": "702-555-1212",
    "LIB_FEAT_ID": 100002,
    "FEAT_DESC_VALUES": [
        {
            "FEAT_DESC": "702-555-1212",
            "LIB_FEAT_ID": 100002
        }
    ]
}
]
```

```

        }
    ],
},
{
    "FEAT_DESC": "702-555-1212",
    "LIB_FEAT_ID": 100002,
    "USAGE_TYPE": "HOME",
    "FEAT_DESC_VALUES": [
        {
            "FEAT_DESC": "702-555-1212",
            "LIB_FEAT_ID": 100002
        }
    ]
},
{
    "FEAT_DESC": "702-555-1313",
    "LIB_FEAT_ID": 100012,
    "USAGE_TYPE": "WORK",
    "FEAT_DESC_VALUES": [
        {
            "FEAT_DESC": "702-555-1313",
            "LIB_FEAT_ID": 100012
        }
    ]
}
],
},
"RECORD_SUMMARY": [
    {
        "DATA_SOURCE": "TEST",
        "RECORD_COUNT": 3
    }
]
},
],
"SEARCH_STATISTICS": [
{
    "CANDIDATE_KEYS": {
        "FEATURE_TYPES": [
            {
                "FTYPE_CODE": "NAME_KEY",
                "FOUND": 2,
                "NOT_FOUND": 0,
                "GENERIC": 0
            }
        ]
    }
}
]

```

```

},
{
    "FTYPE_CODE": "PHONE_KEY",
    "FOUND": 1,
    "NOT_FOUND": 0,
    "GENERIC": 0
},
{
    "FTYPE_CODE": "EMAIL_KEY",
    "FOUND": 1,
    "NOT_FOUND": 0,
    "GENERIC": 0
},
{
    "FTYPE_CODE": "NAMEPHONE_KEY",
    "FOUND": 2,
    "NOT_FOUND": 0,
    "GENERIC": 0
}
],
{
    "SUMMARY": {
        "FOUND": 6,
        "NOT_FOUND": 0,
        "GENERIC": 0
    }
}
}
]
}
}

```

Remarks

The default search profile is "`SEARCH`". Alternatively, "`INGEST`" may be used.

If the specified search profile is `null` then the default will be used (alternatively, use [Search ByAttributes\(string, SzFlag?\)](#) as a convenience method to omit the parameter).

The optionally specified bitwise-OR'd `SzFlag` values not only control how the search is performed but also the content of the response. Any `SzFlag` value may be included, but only flags belonging to the [SzSearchFlags](#) group will be recognized (other `SzFlag` values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzException](#)

If a failure occurs.

See Also

[SearchByAttributes\(string, SzFlag?\)](#), [SzSearchByAttributesDefaultFlags](#),
[SzSearchByAttributesAll](#), [SzSearchByAttributesStrong](#), [SzSearchByAttributesMinimalAll](#),
[SzSearchByAttributesMinimalStrong](#), [SzSearchFlags](#), [Code Snippet: Search Records](#),
[Code Snippet: Search via Futures](#)

WhyEntities(long, long, SzFlag?)

Describes the ways two entities relate to each other.

```
[SzConfigRetryable]  
string WhyEntities(long entityID1, long entityID2, SzFlag? flags =  
SzFlag.SzIncludeFeatureScores)
```

Parameters

entityID1 [long](#)

The entity ID of the first entity.

entityID2 [long](#)

The entity ID of the second entity.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzWhyEntitiesFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter defaults its value to [SzWhyEntitiesDefaultFlags](#) for the default recommended flags. Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) describing the ways in which the records are related to one another.

Examples

Usage:

```

// How to determine how two entities are related
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get the entities on which to operate (varies by application)
    long entityID1 = GetWhyEntitiesID1();
    long entityID2 = GetWhyEntitiesID2();

    // determine how the entities are related
    string results = engine.WhyEntities(entityID1,
                                         entityID2,
                                         SzWhyEntitiesDefaultFlags);

    // do something with the response JSON (varies by application)
    JsonObject? jsonObject = JsonNode.Parse(results)?.AsObject();
    JSONArray? resultsArr = jsonObject?["WHY_RESULTS"]?.AsArray();

    for (int index = 0; resultsArr != null && index < resultsArr.Count; index++)
    {
        JsonObject? result = resultsArr[index]?.AsObject();

        long whyEntityID1 = result?["ENTITY_ID"]?.GetValue<long>() ?? 0L;
        long whyEntityID2 = result?["ENTITY_ID_2"]?.GetValue<long>() ?? 0L;

        . . .

    }
}

catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Entity not found for entity ID.", e);
}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to perform why entities.", e);
}

```

Example Result: (formatted for readability)

```
{  
  "WHY_RESULTS": [  
    {  
      "ENTITY_ID": 5,  
      "ENTITY_ID_2": 8,  
      "MATCH_INFO": {  
        "WHY_KEY": "\u002BPHONE-DOB",  
        "WHY_ERRULE_CODE": "SF1",  
        "MATCH_LEVEL_CODE": "POSSIBLY RELATED",  
        "CANDIDATE_KEYS": {  
          "PHONE_KEY": [  
            {  
              "FEAT_ID": 86,  
              "FEAT_DESC": "8184442121"  
            }  
          ]  
        },  
        "FEATURE_SCORES": {  
          "ADDRESS": [  
            {  
              "INBOUND_FEAT_ID": 3,  
              "INBOUND_FEAT_DESC": "101 Main Street, Los Angeles, CA 90011",  
              "CANDIDATE_FEAT_ID": 161,  
              "CANDIDATE_FEAT_DESC": "707 Seventh Ave, Los Angeles, CA 90043",  
              "SCORE": 30,  
              "ADDITIONAL_SCORES": {  
                "FULL_SCORE": 30  
              },  
              "SCORE_BUCKET": "NO_CHANCE",  
              "SCORE_BEHAVIOR": "FF"  
            }  
          ],  
          "DOB": [  
            {  
              "INBOUND_FEAT_ID": 81,  
              "INBOUND_FEAT_DESC": "22-AUG-1981",  
              "CANDIDATE_FEAT_ID": 160,  
              "CANDIDATE_FEAT_DESC": "27-JUN-1980",  
              "SCORE": 79,  
              "ADDITIONAL_SCORES": {  
                "FULL_SCORE": 79  
              },  
              "SCORE_BUCKET": "NO_CHANCE",  
              "SCORE_BEHAVIOR": "FMES"  
            }  
          ]  
        }  
      }  
    }  
  ]  
}
```

```
        },
    ],
    "NAME": [
        {
            "INBOUND_FEAT_ID": 80,
            "INBOUND_FEAT_DESC": "Bill Bandley",
            "CANDIDATE_FEAT_ID": 159,
            "CANDIDATE_FEAT_DESC": "Katrina Osmond",
            "SCORE": 21,
            "ADDITIONAL_SCORES": {
                "GNR_FN": 21,
                "GNR_SN": 0,
                "GNR_GN": 7,
                "GENERATION_MATCH": -1,
                "GNR_ON": -1
            },
            "SCORE_BUCKET": "NO_CHANCE",
            "SCORE_BEHAVIOR": "NAME"
        }
    ],
    "PHONE": [
        {
            "INBOUND_FEAT_ID": 83,
            "INBOUND_FEAT_DESC": "818-444-2121",
            "INBOUND_FEAT_USAGE_TYPE": "MOBILE",
            "CANDIDATE_FEAT_ID": 83,
            "CANDIDATE_FEAT_DESC": "818-444-2121",
            "CANDIDATE_FEAT_USAGE_TYPE": "MOBILE",
            "SCORE": 100,
            "ADDITIONAL_SCORES": {
                "FULL_SCORE": 100
            },
            "SCORE_BUCKET": "SAME",
            "SCORE_BEHAVIOR": "F1"
        },
        {
            "INBOUND_FEAT_ID": 83,
            "INBOUND_FEAT_DESC": "818-444-2121",
            "INBOUND_FEAT_USAGE_TYPE": "MOBILE",
            "CANDIDATE_FEAT_ID": 162,
            "CANDIDATE_FEAT_DESC": "818-111-2222",
            "CANDIDATE_FEAT_USAGE_TYPE": "HOME",
            "SCORE": 70,
            "ADDITIONAL_SCORES": {
                "FULL_SCORE": 70
            },
            "SCORE_BUCKET": "DIFFERENT"
        }
    ]
}
```

```

        "SCORE_BUCKET": "PLAUSIBLE",
        "SCORE_BEHAVIOR": "FF"
    }
]
},
"DISCLOSED_RELATIONS": {}
}
}
],
"ENTITIES": [
{
    "RESOLVED_ENTITY": {
        "ENTITY_ID": 5
    }
},
{
    "RESOLVED_ENTITY": {
        "ENTITY_ID": 8
    }
}
]
}

```

Remarks

The optionally specified bitwise-OR'd [SzFlag](#) values not only control how the operation is performed but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzWhyEntitiesFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzNotFoundException](#)

If either of the entities for the specified entity ID's could not be found.

[SzException](#)

If a failure occurs.

See Also

[SzWhyEntitiesDefaultFlags](#), [SzWhyEntitiesFlags](#),
[WhyRecords\(string, string, string, string, SzFlag?\)](#),
[WhyRecordInEntity\(string, string, SzFlag?\)](#)

WhyRecordInEntity(string, string, SzFlag?)

Describes the ways a record relates to the rest of its respective entity.

```
[SzConfigRetryable]  
string WhyRecordInEntity(string dataSourceCode, string recordID, SzFlag? flags =  
SzFlag.SzIncludeFeatureScores)
```

Parameters

dataSourceCode [string](#)

The data source code that identifies the data source of the record.

recordID [string](#)

The record ID that identifies the record within the scope of the record's data source.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzWhyRecordInEntityFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter defaults it value to [SzWhyRecordInEntityDefaultFlags](#) for the default recommended flags. Specifying `null` is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) describing why the record is included in its respective entity.

Examples

Usage:

```
// How to determine why a record is a member of its respective entity  
try  
{  
    // obtain the SzEnvironment (varies by application)  
    SzEnvironment env = GetEnvironment();  
  
    // get the engine  
    SzEngine engine = env.GetEngine();
```

```

// determine why the record is part of its entity
string results = engine.WhyRecordInEntity(
    "TEST", "ABC123", SzWhyRecordInEntityDefaultFlags);

// do something with the response JSON (varies by application)
JsonObject? jsonObject = JsonNode.Parse(results)?.AsObject();
JsonArray? resultsArr = jsonObject?["WHY_RESULTS"]?.AsArray();

for (int index = 0; resultsArr != null && index < resultsArr.Count; index++)
{
    JsonObject? result = resultsArr[index]?.AsObject();

    long entityID = result?["ENTITY_ID"]?.GetValue<long>() ?? 0L;

    . . .

}

catch (SzUnknownDataSourceException e)
{
    // handle the unknown data source exception
    LogError("Expected data source is not configured.", e);
}

catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Entity not found for record key.", e);
}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to reevaluate record.", e);
}

```

Example Result: (formatted for readability)

```
{
    "WHY_RESULTS": [
        {
            "INTERNAL_ID": 100002,
            "ENTITY_ID": 100002,
            "FOCUS_RECORDS": [
                {

```

```

    "DATA_SOURCE": "TEST",
    "RECORD_ID": "ABC123"
  }
],
"MATCH_INFO": {
  "WHY_KEY": "\u002BNAME\u002BPHONE\u002BEMAIL",
  "WHY_ERRULE_CODE": "SF1_PNAME_CFF",
  "MATCH_LEVEL_CODE": "RESOLVED",
  "CANDIDATE_KEYS": {
    "EMAIL_KEY": [
      {
        "FEAT_ID": 100007,
        "FEAT_DESC": "joeschmoe@NOWHERE.COM"
      }
    ],
    "NAMEPHONE_KEY": [
      {
        "FEAT_ID": 100008,
        "FEAT_DESC": "JSF|XM|PHONE.PHONE_LAST_5=51212"
      }
    ],
    "NAME_KEY": [
      {
        "FEAT_ID": 100005,
        "FEAT_DESC": "JSF|XM"
      }
    ],
    "PHONE_KEY": [
      {
        "FEAT_ID": 100006,
        "FEAT_DESC": "7025551212"
      }
    ]
  },
  "FEATURE_SCORES": {
    "EMAIL": [
      {
        "INBOUND_FEAT_ID": 100003,
        "INBOUND_FEAT_DESC": "joeschmoe@nowhere.com",
        "CANDIDATE_FEAT_ID": 100003,
        "CANDIDATE_FEAT_DESC": "joeschmoe@nowhere.com",
        "SCORE": 100,
        "ADDITIONAL_SCORES": {
          "FULL_SCORE": 100
        },
        "SCORE_BUCKET": "SAME",
        "CANDIDATE_FEAT_ID": 100003,
        "CANDIDATE_FEAT_DESC": "joeschmoe@nowhere.com"
      }
    ]
  }
}

```

```

        "SCORE_BEHAVIOR": "F1"
    }
],
"NAME": [
{
    "INBOUND_FEAT_ID": 100001,
    "INBOUND_FEAT_DESC": "Joe Schmoe",
    "CANDIDATE_FEAT_ID": 100010,
    "CANDIDATE_FEAT_DESC": "Joseph Schmoe",
    "SCORE": 98,
    "ADDITIONAL_SCORES": {
        "GNR_FN": 98,
        "GNR_SN": -1,
        "GNR_GN": -1,
        "GENERATION_MATCH": -1,
        "GNR_ON": -1
    },
    "SCORE_BUCKET": "CLOSE",
    "SCORE_BEHAVIOR": "NAME"
}
],
"PHONE": [
{
    "INBOUND_FEAT_ID": 100002,
    "INBOUND_FEAT_DESC": "702-555-1212",
    "CANDIDATE_FEAT_ID": 100002,
    "CANDIDATE_FEAT_DESC": "702-555-1212",
    "CANDIDATE_FEAT_USAGE_TYPE": "HOME",
    "SCORE": 100,
    "ADDITIONAL_SCORES": {
        "FULL_SCORE": 100
    },
    "SCORE_BUCKET": "SAME",
    "SCORE_BEHAVIOR": "FF"
}
]
},
"ENTITIES": [
{
    "RESOLVED_ENTITY": {
        "ENTITY_ID": 100002
    }
}
]
,
```

```
]  
}
```

Remarks

The optionally specified bitwise-OR'd [SzFlag](#) values not only control how the operation is performed but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzWhyRecordInEntityFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzUnknownDataSourceException](#)

If an unrecognized data source code is specified.

[SzNotFoundException](#)

If any of the records for the specified data source code and record ID pairs cannot be found.

[SzException](#)

If a failure occurs.

See Also

[SzWhyRecordInEntityDefaultFlags](#), [SzWhyRecordInEntityFlags](#),
[WhyEntities\(long, long, SzFlag?\)](#), [WhyRecords\(string, string, string, string, SzFlag?\)](#).

WhyRecords(string, string, string, string, SzFlag?)

Describes the ways two records relate to each other.

```
[SzConfigRetryable]  
string WhyRecords(string dataSourceCode1, string recordID1, string dataSourceCode2,  
string recordID2, SzFlag? flags = SzFlag.SzIncludeFeatureScores)
```

Parameters

[dataSourceCode1](#) [string](#)

The data source code identifying the data source for the first record.

recordID1 [string](#)

The record ID that uniquely identifies the first record within the scope of its associated data source.

dataSourceCode2 [string](#)

The data source code identifying the data source for the second record.

recordID2 [string](#)

The record ID that uniquely identifies the second record within the scope of its associated data source.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzWhyRecordsFlags](#) group to control how the operation is performed and the content of the response. Omitting this parameter defaults it value to [SzWhyRecordsDefaultFlags](#) for the default recommended flags. Specifying `null` is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The JSON [string](#) describing the ways in which the records are related to one another.

Examples

Usage:

```
// How to determine how two records are related
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get the records on which to operate (varies by application)
    (string dataSourceCode, string recordID) recordKey1 = GetWhyRecordsKey1();
    (string dataSourceCode, string recordID) recordKey2 = GetWhyRecordsKey2();

    // determine how the records are related
```

```

string results = engine.WhyRecords(recordKey1.dataSourceCode,
                                    recordKey1.recordID,
                                    recordKey2.dataSourceCode,
                                    recordKey2.recordID,
                                    SzWhyRecordsDefaultFlags);

// do something with the response JSON (varies by application)
JsonObject? jsonObject = JsonNode.Parse(results)?.AsObject();
JsonArray? resultsArr = jsonObject?["WHY_RESULTS"]?.AsArray();

for (int index = 0; resultsArr != null && index < resultsArr.Count; index++)
{
    JsonObject? result = resultsArr[index]?.AsObject();

    long entityID1 = result?["ENTITY_ID"]?.GetValue<long>() ?? 0L;
    long entityID2 = result?["ENTITY_ID_2"]?.GetValue<long>() ?? 0L;

    . . .

}

}

catch (SzUnknownDataSourceException e)
{
    // handle the unknown data source exception
    LogError("Expected data source is not configured.", e);

}

catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Entity not found for record key.", e);

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to reevaluate record.", e);
}

```

Example Result: (formatted for readability)

```
{
  "WHY_RESULTS": [
    {
      "INTERNAL_ID": 5,

```

```
"ENTITY_ID": 5,
"FOCUS_RECORDS": [
  {
    "DATA_SOURCE": "EMPLOYEES",
    "RECORD_ID": "MNO345"
  }
],
"INTERNAL_ID_2": 8,
"ENTITY_ID_2": 8,
"FOCUS_RECORDS_2": [
  {
    "DATA_SOURCE": "EMPLOYEES",
    "RECORD_ID": "DEF890"
  }
],
"MATCH_INFO": {
  "WHY_KEY": "\u002BPHONE-DOB",
  "WHY_ERRULE_CODE": "SF1",
  "MATCH_LEVEL_CODE": "POSSIBLY RELATED",
  "CANDIDATE_KEYS": {
    "PHONE_KEY": [
      {
        "FEAT_ID": 86,
        "FEAT_DESC": "8184442121"
      }
    ]
  },
  "FEATURE_SCORES": {
    "ADDRESS": [
      {
        "INBOUND_FEAT_ID": 3,
        "INBOUND_FEAT_DESC": "101 Main Street, Los Angeles, CA 90011",
        "CANDIDATE_FEAT_ID": 161,
        "CANDIDATE_FEAT_DESC": "707 Seventh Ave, Los Angeles, CA 90043",
        "SCORE": 30,
        "ADDITIONAL_SCORES": {
          "FULL_SCORE": 30
        },
        "SCORE_BUCKET": "NO_CHANCE",
        "SCORE_BEHAVIOR": "FF"
      }
    ],
    "DOB": [
      {
        "INBOUND_FEAT_ID": 81,
        "INBOUND_FEAT_DESC": "22-AUG-1981",
        "INBOUND_FEAT_TYPE": "DATE"
      }
    ]
  }
}
```

```
"CANDIDATE_FEAT_ID": 160,
"CANDIDATE_FEAT_DESC": "27-JUN-1980",
"SCORE": 79,
"ADDITIONAL_SCORES": {
    "FULL_SCORE": 79
},
"SCORE_BUCKET": "NO_CHANCE",
"SCORE_BEHAVIOR": "FMES"
},
],
"NAME": [
{
    "INBOUND_FEAT_ID": 80,
    "INBOUND_FEAT_DESC": "Bill Bandley",
    "CANDIDATE_FEAT_ID": 159,
    "CANDIDATE_FEAT_DESC": "Katrina Osmond",
    "SCORE": 21,
    "ADDITIONAL_SCORES": {
        "GNR_FN": 21,
        "GNR_SN": 0,
        "GNR_GN": 7,
        "GENERATION_MATCH": -1,
        "GNR_ON": -1
    },
    "SCORE_BUCKET": "NO_CHANCE",
    "SCORE_BEHAVIOR": "NAME"
}
],
"PHONE": [
{
    "INBOUND_FEAT_ID": 83,
    "INBOUND_FEAT_DESC": "818-444-2121",
    "INBOUND_FEAT_USAGE_TYPE": "MOBILE",
    "CANDIDATE_FEAT_ID": 83,
    "CANDIDATE_FEAT_DESC": "818-444-2121",
    "CANDIDATE_FEAT_USAGE_TYPE": "MOBILE",
    "SCORE": 100,
    "ADDITIONAL_SCORES": {
        "FULL_SCORE": 100
    },
    "SCORE_BUCKET": "SAME",
    "SCORE_BEHAVIOR": "F1"
},
{
    "INBOUND_FEAT_ID": 83,
    "INBOUND_FEAT_DESC": "818-444-2121",
```

```

        "INBOUND_FEAT_USAGE_TYPE": "MOBILE",
        "CANDIDATE_FEAT_ID": 162,
        "CANDIDATE_FEAT_DESC": "818-111-2222",
        "CANDIDATE_FEAT_USAGE_TYPE": "HOME",
        "SCORE": 70,
        "ADDITIONAL_SCORES": {
            "FULL_SCORE": 70
        },
        "SCORE_BUCKET": "PLAUSIBLE",
        "SCORE_BEHAVIOR": "FF"
    }
],
},
"DISCLOSED_RELATIONS": {}
}
},
],
"ENTITIES": [
{
    "RESOLVED_ENTITY": {
        "ENTITY_ID": 5
    }
},
{
    "RESOLVED_ENTITY": {
        "ENTITY_ID": 8
    }
}
]
}

```

Remarks

The optionally specified bitwise-OR'd [SzFlag](#) values not only control how the operation is performed but also the content of the response. Any [SzFlag](#) value may be included, but only flags belonging to the [SzWhyRecordsFlags](#) group will be recognized (other [SzFlag](#) values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzUnknownDataSourceException](#)

If an unrecognized data source code is specified.

[SzNotFoundException](#)

If either of the records for the specified data source code and record ID pairs cannot be found.

[SzException](#)

If a failure occurs.

See Also

[SzWhyRecordsDefaultFlags](#), [SzWhyRecordsFlags](#),
[WhyRecordInEntity\(string, string, SzFlag?\)](#), [WhyEntities\(long, long, SzFlag?\)](#)

WhySearch(string, long, string, SzFlag?)

Describes the ways a set of search attributes relate to an entity.

```
[SzConfigRetryable]
string WhySearch(string attributes, long entityID, string searchProfile = null,
SzFlag? flags = SzFlag.SzIncludeFeatureScores | SzFlag.SzSearchIncludeStats |
SzFlag.SzSearchIncludeRequestDetails)
```

Parameters

attributes [string](#)

The search attributes defining the hypothetical record to match and/or relate to in order to obtain the search results.

entityID [long](#)

The entity ID identifying the entity to analyze against the search attribute criteria.

searchProfile [string](#)

The optional search profile identifier, or [null](#) if the default search profile should be used for the search.

flags [SzFlag?](#)

The optional bitwise-OR'd [SzFlag](#) values belonging to the [SzWhySearchFlags](#) group to control how the operation is performed and the content of the response, omitting this parameter will default its value to [SzWhySearchDefaultFlags](#) for the default recommended flags. Specifying [null](#) is equivalent to specifying [SzNoFlags](#).

Returns

[string](#)

The resulting JSON `string` describing the result of the why analysis against the search criteria.

Examples

Usage:

```
// How to determine why an entity was excluded from search results
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    // get the search attributes (varies by application)
    string searchAttributes =
        """
        {
            "NAME_FULL": "Joe Schmoe",
            "PHONE_NUMBER": "702-555-1212",
            "EMAIL_ADDRESS": "joeschmoe@nowhere.com"
        }
        """;
}

// get the entities on which to operate (varies by application)
long entityID = GetWhySearchEntityID();

// determine how the entities are related
string results = engine.WhySearch(searchAttributes,
                                    entityID,
                                    null, // search profile
                                    SzWhySearchDefaultFlags);

// do something with the response JSON (varies by application)
JsonObject? jsonObject = JsonNode.Parse(results)?.AsObject();
JsonArray? resultsArr = jsonObject?["WHY_RESULTS"]?.AsArray();

for (int index = 0; resultsArr != null && index < resultsArr.Count; index++)
{
```

```

JsonObject? result = resultsArr[index]?.AsObject();

long whyEntityID1 = result?["ENTITY_ID"]?.GetValue<long>() ?? 0L;

. . .

}

}

catch (SzNotFoundException e)
{
    // handle the not-found exception
    LogError("Entity not found for entity ID.", e);

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to perform why search.", e);
}

```

Example Result: (formatted for readability)

```
{
    "WHY_RESULTS": [
        {
            "ENTITY_ID": 5,
            "MATCH_INFO": {
                "WHY_KEY": "",
                "WHY_ERRULE_CODE": "",
                "MATCH_LEVEL_CODE": "",
                "CANDIDATE_KEYS": {},
                "FEATURE_SCORES": {},
                "DISCLOSED_RELATIONS": {}
            }
        }
    ],
    "SEARCH_REQUEST": {
        "JSON_DATA": "{\n            \"NAME_FULL\": \"Joe Schmoe\", \n            \"PHONE_NUMBER\": \"702-555-1212\", \n            \"EMAIL_ADDRESS\": \"joeschmoe@nowhere.com\"\n        }",
        "SEARCH_PROFILE": "",
        "FEATURES": {
            "NAME": [
                {
                    "LIB_FEAT_ID": 100001,

```

```
"FEAT_DESC": "Joe Schmoe",
"ATTRIBUTES": {
    "NAME_FULL": "Joe Schmoe"
},
"USED_FOR_CAND": "N",
"USED_FOR_SCORING": "Y",
"ENTITY_COUNT": 1,
"CANDIDATE_CAP_REACHED": "N",
"SCORING_CAP_REACHED": "N"
}
],
"PHONE": [
{
    "LIB_FEAT_ID": 100002,
    "FEAT_DESC": "702-555-1212",
    "ATTRIBUTES": {
        "PHONE_NUMBER": "702-555-1212"
    },
    "USED_FOR_CAND": "N",
    "USED_FOR_SCORING": "Y",
    "ENTITY_COUNT": 1,
    "CANDIDATE_CAP_REACHED": "N",
    "SCORING_CAP_REACHED": "N"
}
],
"EMAIL": [
{
    "LIB_FEAT_ID": 100003,
    "FEAT_DESC": "joeschmoe@nowhere.com",
    "ATTRIBUTES": {
        "EMAIL_ADDRESS": "joeschmoe@nowhere.com"
    },
    "USED_FOR_CAND": "N",
    "USED_FOR_SCORING": "Y",
    "ENTITY_COUNT": 1,
    "CANDIDATE_CAP_REACHED": "N",
    "SCORING_CAP_REACHED": "N"
}
],
"NAME_KEY": [
{
    "LIB_FEAT_ID": 100004,
    "FEAT_DESC": "JOE|XM",
    "USED_FOR_CAND": "Y",
    "USED_FOR_SCORING": "N",
    "ENTITY_COUNT": 1,
```

```
"CANDIDATE_CAP_REACHED": "N",
"SCORING_CAP_REACHED": "N"
},
{
  "LIB_FEAT_ID": 100005,
  "FEAT_DESC": "JSF|XM",
  "USED_FOR_CAND": "Y",
  "USED_FOR_SCORING": "N",
  "ENTITY_COUNT": 1,
  "CANDIDATE_CAP_REACHED": "N",
  "SCORING_CAP_REACHED": "N"
}
],
"PHONE_KEY": [
{
  "LIB_FEAT_ID": 100006,
  "FEAT_DESC": "7025551212",
  "USED_FOR_CAND": "Y",
  "USED_FOR_SCORING": "N",
  "ENTITY_COUNT": 1,
  "CANDIDATE_CAP_REACHED": "N",
  "SCORING_CAP_REACHED": "N"
}
],
"EMAIL_KEY": [
{
  "LIB_FEAT_ID": 100007,
  "FEAT_DESC": "joeschmoe@NOWHERE.COM",
  "USED_FOR_CAND": "Y",
  "USED_FOR_SCORING": "N",
  "ENTITY_COUNT": 1,
  "CANDIDATE_CAP_REACHED": "N",
  "SCORING_CAP_REACHED": "N"
}
],
"NAMEPHONE_KEY": [
{
  "LIB_FEAT_ID": 100008,
  "FEAT_DESC": "JSF|XM|PHONE.PHONE_LAST_5=51212",
  "USED_FOR_CAND": "Y",
  "USED_FOR_SCORING": "N",
  "ENTITY_COUNT": 1,
  "CANDIDATE_CAP_REACHED": "N",
  "SCORING_CAP_REACHED": "N"
}
],
{
```

```
"LIB_FEAT_ID": 100009,
"FEAT_DESC": "JOE|XM|PHONE.PHONE_LAST_5=51212",
"USED_FOR_CAND": "Y",
"USED_FOR_SCORING": "N",
"ENTITY_COUNT": 1,
"CANDIDATE_CAP_REACHED": "N",
"SCORING_CAP_REACHED": "N"
}
]
}
},
"SEARCH_STATISTICS": [
{
"CANDIDATE_KEYS": {
"FEATURE_TYPES": [
{
"FTYPE_CODE": "NAME_KEY",
"FOUND": 0,
"NOT_FOUND": 2,
"GENERIC": 0
},
{
"FTYPE_CODE": "PHONE_KEY",
"FOUND": 0,
"NOT_FOUND": 1,
"GENERIC": 0
},
{
"FTYPE_CODE": "EMAIL_KEY",
"FOUND": 0,
"NOT_FOUND": 1,
"GENERIC": 0
},
{
"FTYPE_CODE": "NAMEPHONE_KEY",
"FOUND": 0,
"NOT_FOUND": 2,
"GENERIC": 0
}
],
"SUMMARY": {
"FOUND": 0,
"NOT_FOUND": 6,
"GENERIC": 0
}
}
}
```

```
        }
    ],
    "ENTITIES": [
        {
            "RESOLVED_ENTITY": {
                "ENTITY_ID": 5
            }
        }
    ]
}
```

Remarks

The default search profile is "`SEARCH`". Alternatively, "`INGEST`" may be used.

If the specified search profile is `null` then the default will be used.

The specified search attributes are treated as a hypothetical single-record entity and the result of this operation is the "why analysis" of the entity identified by the specified entity ID against that hypothetical entity. The details included in the response are determined by the specified flags.

If the specified search profile is `null` then the default generic thresholds from the default search profile will be used for the search candidate determination. If your search requires different behavior using alternate generic thresholds, please contact support@senzing.com ↗ for details on configuring a custom search profile.

The optionally specified bitwise-OR'd `SzFlag` values not only control how the search is performed but also the content of the response. Any `SzFlag` value may be included, but only flags belonging to the `SzWhySearchFlags` group will be recognized (other `SzFlag` values will be ignored unless they have equivalent bit flags to recognized flags).

Exceptions

[SzNotFoundException](#)

If no entity could be found with the specified entity ID.

[SzException](#)

If a failure occurs.

See Also

[SzWhySearchDefaultFlags](#), [SzWhySearchFlags](#)

Interface SzEnvironment

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Provides a factory interface for obtaining the references to the Senzing SDK singleton instances that have been initialized.

```
public interface SzEnvironment
```

Examples

Usage:

```
// get the settings (varies by application)
string settings = GetSettings();

// get the instance name (varies by application)
string instanceName = GetInstanceName();

// construct the environment
SzEnvironment env = SzCoreEnvironment.NewBuilder()
    .InstanceId(instanceId)
    .Settings(settings)
    .VerboseLogging(false)
    .Build();

// use the environment for some time (usually as long as the application is running)
. . .

// destroy the environment when done (sometimes in a finally block)
env.Destroy();
```

Methods

Destroy()

Destroys this **SzEnvironment** and invalidates any SDK singleton references that has previously provided.

```
void Destroy()
```

Examples

Usage:

```
// obtain the SzEnvironment (varies by application)
SzEnvironment env = GetEnvironment();

// check if already destroyed
if (!env.IsDestroyed())
{
    // destroy the environment
    env.Destroy();
}
```

Remarks

If this instance has already been destroyed then this method has no effect.

GetActiveConfigID()

Gets the currently active configuration ID for this [SzEnvironment](#).

```
long GetActiveConfigID()
```

Returns

[long](#)

The currently active configuration ID.

Examples

Usage:

```
// How to get the active config ID for the SzEnvironment
try
{
    // obtain the SzEnvironment (varies by application)
```

```

SzEnvironment env = GetEnvironment();

// get the active config ID
long activeConfigID = env.GetActiveConfigID();

// do something with the active config ID (varies by application)
SzConfigManager configMgr = env.GetConfigManager();

long defaultConfigID = configMgr.GetDefaultConfigID();

if (activeConfigID != defaultConfigID)
{
    // reinitialize the environment with the default config ID
    env.Reinitialize(defaultConfigID);
}

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to verify the active config ID.", e);
}

```

Exceptions

[SzEnvironmentDestroyedException](#)

If this `SzEnvironment` instance has been destroyed.

[SzException](#)

If there was a failure in obtaining the active config ID.

GetConfigManager()

Provides a reference to the [SzConfigManager](#) instance associated with this `SzEnvironment`.

`SzConfigManager GetConfigManager()`

Returns

[SzConfigManager](#)

The [SzConfigManager](#) instance associated with this [SzEnvironment](#).

Examples

Usage:

```
// How to obtain an SzConfigManager instance
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the config manager
    SzConfigManager configMgr = env.GetConfigManager();

    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to get SzConfigManager.", e);
}
```

Exceptions

[SzEnvironmentDestroyedException](#)

If this [SzEnvironment](#) instance has been destroyed.

[SzException](#)

If there was a failure in obtaining or initializing the [SzConfigManager](#) instance.

GetDiagnostic()

Provides a reference to the [SzDiagnostic](#) instance associated with this [SzEnvironment](#).

[SzDiagnostic](#) `GetDiagnostic()`

Returns

[SzDiagnostic](#)

The [SzDiagnostic](#) instance associated with this [SzEnvironment](#).

Examples

Usage:

```
// How to obtain an SzDiagnostic instance
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    SzDiagnostic diagnostic = env.GetDiagnostic();

    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to get SzDiagnostic.", e);
}
```

Exceptions

[SzEnvironmentDestroyedException](#)

If this [SzEnvironment](#) instance has been destroyed.

[SzException](#)

If there was a failure in obtaining or initializing the [SzDiagnostic](#) instance.

GetEngine()

Provides a reference to the [SzEngine](#) instance associated with this [SzEnvironment](#).

SzEngine [GetEngine\(\)](#)

Returns

[SzEngine](#)

The [SzEngine](#) instance associated with this [SzEnvironment](#).

Examples

Usage:

```
// How to obtain an SzEngine instance
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the engine
    SzEngine engine = env.GetEngine();

    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to get SzEngine.", e);
}
```

Exceptions

[SzEnvironmentDestroyedException](#)

If this [SzEnvironment](#) instance has been destroyed.

[SzException](#)

If there was a failure in obtaining or initializing the [SzEngine](#) instance.

GetProduct()

Provides a reference to the [SzProduct](#) singleton associated with this [SzEnvironment](#).

SzProduct [GetProduct\(\)](#)

Returns

[SzProduct](#)

The [SzProduct](#) instance associated with this [SzEnvironment](#)

Examples

Usage:

```
// How to obtain an SzProduct instance
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the product from the environment
    SzProduct product = env.GetProduct();

    . . .

}
catch (SzException e)
{
    // handle or rethrow the exception (varies by application)
    LogError("Failed to get SzProduct.", e);
}
```

Exceptions

[SzEnvironmentDestroyedException](#)

If this [SzEnvironment](#) instance has been [destroyed](#).

[SzException](#)

If there was a failure in obtaining or initializing the [SzProduct](#) instance.

[IsDestroyed\(\)](#)

Checks if this instance has had its [Destroy\(\)](#) method called.

```
bool IsDestroyed()
```

Returns

[bool](#)

`true` if this instance has had its [Destroy\(\)](#) method called, otherwise `false`.

Examples

Usage:

```
// obtain the SzEnvironment (varies by application)
SzEnvironment env = GetEnvironment();

// check if already destroyed
if (!env.IsDestroyed())
{
    // destroy the environment
    env.Destroy();
}
```

Reinitialize(long)

Reinitializes the `SzEnvironment` with the specified configuration ID.

```
void Reinitialize(long configID)
```

Parameters

`configID` [long](#)

The configuration ID with which to initialize.

Examples

Usage:

```

// How to get the active config ID for the SzEnvironment
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the active config ID
    long activeConfigID = env.GetActiveConfigID();

    // do something with the active config ID (varies by application)
    SzConfigManager configMgr = env.GetConfigManager();

    long defaultConfigID = configMgr.GetDefaultConfigID();

    if (activeConfigID != defaultConfigID)
    {
        // reinitialize the environment with the default config ID
        env.Reinitialize(defaultConfigID);
    }
}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to verify the active config ID.", e);
}

```

Exceptions

[SzEnvironmentDestroyedException](#)

If this `SzEnvironment` instance has been destroyed.

[SzException](#)

If there was a failure reinitializing.

Class SzEnvironmentDestroyedException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Extends [InvalidOperationException](#) so the exceptional condition of the [SzEnvironment](#) already being destroyed can be differentiated from other [InvalidOperationException](#) instances that might be thrown.

```
public class SzEnvironmentDestroyedException : InvalidOperationException,  
ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SystemException](#) ← [InvalidOperationException](#) ←
SzEnvironmentDestroyedException

Implements

[ISerializable](#)

Inherited Members

[Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Remarks

This exception can be thrown from almost all Senzing SDK functions if the associated [SzEnvironment](#) has been destroyed.

NOTE: This class does **not** extend [SzException](#) but rather extends [InvalidOperationException](#).

Constructors

[SzEnvironmentDestroyedException\(\)](#)

Default constructor.

```
public SzEnvironmentDestroyedException()
```

SzEnvironmentDestroyedException(Exception)

Constructs with the specified [Exception](#) cause.

```
public SzEnvironmentDestroyedException(Exception cause)
```

Parameters

cause [Exception](#)

The [Exception](#) cause

SzEnvironmentDestroyedException(string)

Constructs with the specified message.

```
public SzEnvironmentDestroyedException(string message)
```

Parameters

message [string](#)

The message describing what occurred.

SzEnvironmentDestroyedException(string, Exception)

Constructs with the specified message and [Exception](#) cause.

```
public SzEnvironmentDestroyedException(string message, Exception cause)
```

Parameters

message [string](#)

The message describing what occurred.

cause [Exception](#)

The [Exception](#) cause

Class SzException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Defines the base exception for Senzing errors. This adds a property for the numeric Senzing error code which can optionally be set.

```
public class SzException : Exception, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← SzException

Implements

[ISerializable](#)

Derived

[SzBadInputException](#), [SzConfigurationException](#), [SzReplaceConflictException](#),
[SzRetryableException](#), [SzUnrecoverableException](#)

Inherited Members

[Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

SzException()

Default constructor.

```
public SzException()
```

SzException(Exception)

Constructs with the [Exception](#) that is the underlying cause for the exception.

```
public SzException(Exception cause)
```

Parameters

cause [Exception](#)

The underlying cause for the exception.

SzException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzException(long? errorCode, string message, Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzException(string)

Constructs with a message explaining the reason for the exception.

```
public SzException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Properties

ErrorCode

Gets the underlying Senzing error code associated with the exception or `null` if no error code was associated with the exception.

```
public long? ErrorCode { get; }
```

Property Value

[long](#)?

Enum SzFlag

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Enumerates the Senzing flag values so they can be referred to as bitwise enumerated flags.

```
[Flags]
public enum SzFlag : long
```

Extension Methods

[SzFlags.FlagsToString\(SzFlag\)](#)

Fields

SzEntityIncludeAllFeatures = 1024

The bitwise flag for including all features for entities.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzEntityIncludeDisclosedRelations = 512

The bitwise flag for including disclosed relations for entities.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)

- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzEntityIncludeEntityName = 4096

The bitwise flag for including the name of the entity.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzEntityIncludeFeatureStats = 16777216

The bitwise flag for including feature statistics in entity responses.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzEntityIncludeInternalFeatures = 8388608

The bitwise flag for including internal features in an entity response or record response.

This flag belongs to the following usage groups:

- [SzRecordPreviewFlags](#)
- [SzRecordFlags](#)
- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzEntityIncludeNameOnlyRelations = 256

The bitwise flag for including name-only relations for entities.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzEntityIncludePossiblyRelatedRelations = 128

The bitwise flag for including possibly-related relations for entities.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)

- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludePossiblySameRelations = 64

The bitwise flag for including possibly-same relations for entities.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludeRecordData = 16384

The bitwise flag for including the basic record data for the entity.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludeRecordDates = 549755813888

The bitwise flag for including the record matching info for the entity or record.

This flag belongs to the following usage groups:

- [SzRecordFlags](#)
- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludeRecordFeatureDetails = 34359738368

The bitwise flag for including full feature details at the record level of an entity response or in a record response.

This flag belongs to the following usage groups:

- [SzRecordPreviewFlags](#)
- [SzRecordFlags](#)
- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludeRecordFeatureStats = 68719476736

The bitwise flag for including full feature statistics at the record level of an entity response or in a record response.

This flag belongs to the following usage groups:

- [SzRecordPreviewFlags](#)
- [SzRecordFlags](#)
- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludeRecordFeatures = 262144

The bitwise flag to include the features identifiers at the record level, referencing the entity features.

This flag belongs to the following usage groups:

- [SzRecordPreviewFlags](#)
- [SzRecordFlags](#)
- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludeRecordJsonData = 65536

The bitwise flag for including the record json data for the entity or record.

This flag belongs to the following usage groups:

- [SzRecordPreviewFlags](#)

- [SzRecordFlags](#)
- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludeRecordMatchingInfo = 32768

The bitwise flag for including the record matching info for the entity or record.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludeRecordSummary = 8192

The bitwise flag for including the record summary of the entity.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)

- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzEntityIncludeRecordTypes = 268435456

The bitwise flag for including the record types of the entity or record.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzEntityIncludeRecordUnmappedData = 2147483648

The bitwise flag for including the record unmapped data for the entity or record.

This flag belongs to the following usage groups:

- [SzRecordPreviewFlags](#)
- [SzRecordFlags](#)
- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzEntityIncludeRelatedEntityName = 524288

The bitwise flag for including the name of the related entities.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludeRelatedMatchingInfo = 1048576

The bitwise flag for including the record matching info of the related entities.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludeRelatedRecordData = 4194304

The bitwise flag for including the basic record data of the related entities.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)

- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludeRelatedRecordSummary = 2097152

The bitwise flag for including the record summary of the related entities.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludeRelatedRecordTypes = 536870912

The bitwise flag for including the record types of the related entities.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityIncludeRepresentativeFeatures = 2048

The bitwise flag for including representative features for entities.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzVirtualEntityFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzExportIncludeDisclosed = 16

The bitwise flag for export functionality to indicate that we should include "disclosed" relationships.

This flag belongs to the following usage groups:

- [SzExportFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzExportIncludeMultiRecordEntities = 1

The bitwise flag for export functionality to indicate "resolved" relationships (i.e.: entities with multiple records) should be included in the export.

This flag belongs to the following usage groups:

- [SzExportFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzExportIncludeNameOnly = 8

The bitwise flag for export functionality to indicate that "name only" relationships should be included in the export.

This flag belongs to the following usage groups:

- [SzExportFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzExportIncludePossiblyRelated = 4

The bitwise flag for export functionality to indicate that "possibly related" relationships should be included in the export.

This flag belongs to the following usage groups:

- [SzExportFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzExportIncludePossiblySame = 2

The bitwise flag for export functionality to indicate that "possibly same" relationships should be included in the export.

This flag belongs to the following usage groups:

- [SzExportFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzExportIncludeSingleRecordEntities = 32

The bitwise flag for export functionality to indicate that single-record entities should be included in the export.

This flag belongs to the following usage groups:

- [SzExportFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzFindNetworkIncludeMatchingInfo = 8589934592

The bitwise flag for find-network functionality to include matching info on entity paths of the network.

This flag belongs to the following usage groups:

- [SzFindNetworkFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

SzFindPathIncludeMatchingInfo = 1073741824

The bitwise flag for find-path functionality to include matching info on entity paths.

This flag belongs to the following usage groups:

- [SzFindPathFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzFindPathStrictAvoid = 33554432

The bitwise flag for find-path functionality to indicate that avoided entities are not allowed under any circumstance -- even if they are the only means by which a path can be found between two entities.

This flag belongs to the following usage groups:

- [SzFindPathFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzIncludeFeatureScores = 67108864

The bitwise flag for including feature scores.

This flag belongs to the following usage groups:

- [SzSearchFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)
- [SzWhySearchFlags](#)
- [SzHowFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzIncludeMatchKeyDetails = 17179869184

The bitwise flag for including match key details in addition to the standard match key.

This flag belongs to the following usage groups:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

- [SzHowFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzSearchIncludeAllCandidates = 4294967296

The bitwise flag for search functionality to indicate that search results should not only include those entities that satisfy resolution rule, but also those that present on the candidate list but fail to satisfy a resolution rule.

This flag belongs to the following usage groups:

- [SzSearchFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzSearchIncludeNameOnly = 8

The bitwise flag for search functionality to indicate that "name only" match level results should be included.

This flag belongs to the following usage groups:

- [SzSearchFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzSearchIncludePossiblyRelated = 4

The bitwise flag for search functionality to indicate that "possibly related" match level results should be included.

This flag belongs to the following usage groups:

- [SzSearchFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzSearchIncludePossiblySame = 2

The bitwise flag for search functionality to indicate that "possibly same" match level results should be included.

This flag belongs to the following usage groups:

- [SzSearchFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzSearchIncludeRequest = 137438953472

The bitwise flag for search functionality to indicate that the search response should include the basic feature information for the search criteria features.

This flag belongs to the following usage groups:

- [SzSearchFlags](#)
- [SzWhySearchFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzSearchIncludeRequestDetails = 274877906944

The bitwise flag for search functionality to indicate that the search response should include detailed feature information for search criteria features (including feature stats and generic status).

This flag has no effect unless [SzSearchIncludeRequest](#) is also specified.

This flag belongs to the following usage groups:

- [SzSearchFlags](#)
- [SzWhySearchFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzSearchIncludeResolved = 1

The bitwise flag for search functionality to indicate that "resolved" match level results should be included.

This flag belongs to the following usage groups:

- [SzSearchFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzSearchIncludeStats = 134217728

The bitwise flag for including statistics from search results.

This flag belongs to the following usage groups:

- [SzSearchFlags](#)

- [SzWhySearchFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzWithInfo = 4611686018427387904

The bitwise flag that indicates that the Senzing engine should produce and return the INFO document describing the affected entities from an operation.

This flag belongs to the following usage groups:

- [SzAddRecordFlags](#)
- [SzDeleteRecordFlags](#)
- [SzReevaluateRecordFlags](#)
- [SzReevaluateEntityFlags](#)
- [SzRedoFlags](#)

Remarks

Each [SzFlag](#) belongs to one or more [SzFlagUsageGroup](#) instances which can be obtained via the [GetGroups\(string\)](#) extension method. This helps in identifying which flags are applicable to which functions since a function will document which [SzFlagUsageGroup](#) to refer to for applicable flags.

Passing an [SzFlag](#) to a function to which it does not apply will either have no effect or activate an applicable [SzFlag](#) that happens to have the same bitwise value as the non-applicable [SzFlag](#).

See Also

<https://docs.senzing.com/flags/index.html>

Enum SzFlagUsageGroup

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Enumerates the various classifications of usage groups for the [SzFlag](#) instances.

```
[Flags]
public enum SzFlagUsageGroup : long
```

Extension Methods

[SzFlags.FlagsToString\(SzFlagUsageGroup, SzFlag?\)](#)

Fields

SzAddRecordFlags = 1

Flags in this usage group can be used for operations that add (load) records to the entity repository.

Applicable methods include:

- [AddRecord\(string, string, string, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzWithInfo](#)

The pre-defined [SzFlag](#) aggregate constants for this group are:

- [SzAddRecordDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzDeleteRecordFlags = 2

Flags in this usage group can be used for operations that delete records from the entity repository.

Applicable methods include:

- [DeleteRecord\(string, string, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzWithInfo](#)

The pre-defined [SzFlag](#) aggregate constants for this group are:

- [SzDeleteRecordDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzEntityFlags = 128

Flags in this usage group can be used for operations that retrieve entity data in order to control the level of detail of the returned entity.

Applicable methods include:

- [GetEntity\(long, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzEntityIncludePossiblySameRelations](#)
- [SzEntityIncludePossiblyRelatedRelations](#)
- [SzEntityIncludeNameOnlyRelations](#)
- [SzEntityIncludeDisclosedRelations](#)
- [SzEntityIncludeAllFeatures](#)
- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)
- [SzEntityIncludeRecordTypes](#)
- [SzEntityIncludeRecordData](#)
- [SzEntityIncludeRecordMatchingInfo](#)
- [SzEntityIncludeRecordJsonData](#)
- [SzEntityIncludeRecordUnmappedData](#)
- [SzEntityIncludeRecordFeatures](#)
- [SzEntityIncludeRecordFeatureDetails](#)
- [SzEntityIncludeRecordFeatureStats](#)
- [SzEntityIncludeRelatedEntityName](#)
- [SzEntityIncludeRelatedMatchingInfo](#)
- [SzEntityIncludeRelatedRecordSummary](#)
- [SzEntityIncludeRelatedRecordTypes](#)
- [SzEntityIncludeRelatedRecordData](#)
- [SzEntityIncludeInternalFeatures](#)
- [SzEntityIncludeFeatureStats](#)
- [SzIncludeMatchKeyDetails](#)

The pre-defined [SzFlag](#) aggregate constants for this group are:

- [SzEntityIncludeAllRelations](#)
- [SzEntityDefaultFlags](#)
- [SzEntityBriefDefaultFlags](#)

The pre-defined [SzFlag](#) aggregate constants that also support this group for defining entity or record detail levels are:

- [SzRecordDefaultFlags](#)
- [SzVirtualEntityDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzExportFlags = 4096

Flags in this usage group can be used for operations that export entities to control how the entities are qualified for inclusion in the export and the level of detail for the entities returned in the search results.

Applicable methods include:

- [ExportJsonEntityReport\(SzFlag?\)](#)
- [ExportCsvEntityReport\(string, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzEntityIncludePossiblySameRelations](#)
- [SzEntityIncludePossiblyRelatedRelations](#)
- [SzEntityIncludeNameOnlyRelations](#)
- [SzEntityIncludeDisclosedRelations](#)
- [SzEntityIncludeAllFeatures](#)
- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)
- [SzEntityIncludeRecordTypes](#)
- [SzEntityIncludeRecordData](#)
- [SzEntityIncludeRecordMatchingInfo](#)
- [SzEntityIncludeRecordJsonData](#)
- [SzEntityIncludeRecordUnmappedData](#)
- [SzEntityIncludeRecordFeatures](#)
- [SzEntityIncludeRecordFeatureDetails](#)
- [SzEntityIncludeRecordFeatureStats](#)
- [SzEntityIncludeRelatedEntityName](#)

- [SzEntityIncludeRelatedMatchingInfo](#)
- [SzEntityIncludeRelatedRecordSummary](#)
- [SzEntityIncludeRelatedRecordTypes](#)
- [SzEntityIncludeRelatedRecordData](#)
- [SzEntityIncludeInternalFeatures](#)
- [SzEntityIncludeFeatureStats](#)
- [SzExportIncludeMultiRecordEntities](#)
- [SzExportIncludePossiblySame](#)
- [SzExportIncludePossiblyRelated](#)
- [SzExportIncludeNameOnly](#)
- [SzExportIncludeDisclosed](#)
- [SzExportIncludeSingleRecordEntities](#)

The pre-defined [SzFlag](#) aggregate constants that use this group and are defined for "export" operations are:

- [SzExportIncludeAllEntities](#)
- [SzExportIncludeAllHavingRelationships](#)
- [SzExportDefaultFlags](#)

The pre-defined [SzFlag](#) aggregate constants that also support this group for defining entity or record detail levels are:

- [SzRecordDefaultFlags](#)
- [SzEntityIncludeAllRelations](#)
- [SzEntityDefaultFlags](#)
- [SzEntityBriefDefaultFlags](#)
- [SzVirtualEntityDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzFindInterestingEntitiesFlags = 1024

Flags in this usage group can be used to control the methodology for finding an interesting entities and what details to include for the results of the operation.

Applicable methods include:

- [FindInterestingEntities\(string, string, SzFlag?\)](#)
- [FindInterestingEntities\(long, SzFlag?\)](#)

Currently, there are no [SzFlag](#) instances included in this usage group. However, this is a placeholder for when such flags are defined in a future release.

The pre-defined [SzFlag](#) aggregate constants that use this group and are defined for "find-path" operations are:

- [SzFindInterestingEntitiesDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

`SzFindNetworkFlags = 512`

Flags in this usage group can be used to control the methodology for finding an entity network, what details to include for the entity network and the level of detail for the entities in the network that are returned.

Applicable methods include:

- [FindNetwork\(ISet<long>, int, int, int, SzFlag?\)](#)
- [FindNetwork\(ISet<\(string dataSourceCode, string recordID\)>, int, int, int, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzEntityIncludePossiblySameRelations](#)
- [SzEntityIncludePossiblyRelatedRelations](#)
- [SzEntityIncludeNameOnlyRelations](#)
- [SzEntityIncludeDisclosedRelations](#)
- [SzEntityIncludeAllFeatures](#)
- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)
- [SzEntityIncludeRecordTypes](#)
- [SzEntityIncludeRecordData](#)
- [SzEntityIncludeRecordMatchingInfo](#)
- [SzEntityIncludeRecordJsonData](#)
- [SzEntityIncludeRecordUnmappedData](#)
- [SzEntityIncludeRecordFeatures](#)
- [SzEntityIncludeRecordFeatureDetails](#)
- [SzEntityIncludeRecordFeatureStats](#)
- [SzEntityIncludeRelatedEntityName](#)
- [SzEntityIncludeRelatedMatchingInfo](#)
- [SzEntityIncludeRelatedRecordSummary](#)
- [SzEntityIncludeRelatedRecordTypes](#)
- [SzEntityIncludeRelatedRecordData](#)
- [SzEntityIncludeInternalFeatures](#)
- [SzIncludeMatchKeyDetails](#)

- [SzEntityIncludeFeatureStats](#)
- [SzFindNetworkIncludeMatchingInfo](#)

The pre-defined [SzFlag](#) aggregate constants that use this group and are defined for "find-path" operations are:

- [SzFindNetworkDefaultFlags](#)

The pre-defined [SzFlag](#) aggregate constants that also support this group for defining entity or record detail levels are:

- [SzRecordDefaultFlags](#)
- [SzEntityIncludeAllRelations](#)
- [SzEntityDefaultFlags](#)
- [SzEntityBriefDefaultFlags](#)
- [SzVirtualEntityDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzFindPathFlags = 256

Flags in this usage group can be used to control the methodology for finding an entity path, what details to include for the entity path and the level of detail for the entities on the path that are returned.

Applicable methods include:

- [FindPath\(long, long, int, ISet<long>, ISet<string>, SzFlag?\)](#).
- [FindPath\(string, string, string, string, int, ISet<\(string dataSourceCode, string recordID\)>, ISet<string>, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzEntityIncludePossiblySameRelations](#)
- [SzEntityIncludePossiblyRelatedRelations](#)
- [SzEntityIncludeNameOnlyRelations](#)
- [SzEntityIncludeDisclosedRelations](#)
- [SzEntityIncludeAllFeatures](#)
- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)
- [SzEntityIncludeRecordTypes](#)
- [SzEntityIncludeRecordData](#)
- [SzEntityIncludeRecordMatchingInfo](#)

- [SzEntityIncludeRecordJsonData](#)
- [SzEntityIncludeRecordUnmappedData](#)
- [SzEntityIncludeRecordFeatures](#)
- [SzEntityIncludeRecordFeatureDetails](#)
- [SzEntityIncludeRecordFeatureStats](#)
- [SzEntityIncludeRelatedEntityName](#)
- [SzEntityIncludeRelatedMatchingInfo](#)
- [SzEntityIncludeRelatedRecordSummary](#)
- [SzEntityIncludeRelatedRecordTypes](#)
- [SzEntityIncludeRelatedRecordData](#)
- [SzEntityIncludeInternalFeatures](#)
- [SzEntityIncludeFeatureStats](#)
- [SzIncludeMatchKeyDetails](#)
- [SzFindPathStrictAvoid](#)
- [SzFindPathIncludeMatchingInfo](#)

The pre-defined [SzFlag](#) aggregate constants that use this group and are defined for "find-path" operations are:

- [SzFindPathDefaultFlags](#)

The pre-defined [SzFlag](#) aggregate constants that also support this group for defining entity or record detail levels are:

- [SzRecordDefaultFlags](#)
- [SzEntityIncludeAllRelations](#)
- [SzEntityDefaultFlags](#)
- [SzEntityBriefDefaultFlags](#)
- [SzVirtualEntityDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzHowFlags = 131072

Flags in this usage group can be used to control the methodology for performing "how analysis", what details to include for the analysis and the level of detail for the entities in the network that are returned.

Applicable methods include:

- [HowEntity\(long, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzIncludeMatchKeyDetails](#)
- [SzIncludeFeatureScores](#)

The pre-defined [SzFlag](#) aggregate constants that use this group and are defined for "how" operations are:

- [SzHowEntityDefaultFlags](#)

The pre-defined [SzFlag](#) aggregate constants that also support this group for defining entity or record detail levels are:

- [SzRecordDefaultFlags](#)
- [SzEntityIncludeAllRelations](#)
- [SzEntityDefaultFlags](#)
- [SzEntityBriefDefaultFlags](#)
- [SzVirtualEntityDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzRecordFlags = 32

Flags in this usage group can be used for operations that retrieve record data in order to control the level of detail of the returned record.

Applicable methods include:

- [GetRecord\(string, string, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzEntityIncludeInternalFeatures](#)
- [SzEntityIncludeRecordFeatures](#)
- [SzEntityIncludeRecordFeatureDetails](#)
- [SzEntityIncludeRecordFeatureStats](#)
- [SzEntityIncludeRecordDates](#)
- [SzEntityIncludeRecordJsonData](#)
- [SzEntityIncludeRecordUnmappedData](#)

The pre-defined [SzFlag](#) aggregate constants for this group are:

- [SzRecordDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzRecordPreviewFlags = 64

Flags in this usage group can be used for operations that retrieve record data in order to control the level of detail of the returned record.

Applicable methods include:

- [GetRecord\(string, string, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzEntityIncludeInternalFeatures](#)
- [SzEntityIncludeRecordFeatures](#)
- [SzEntityIncludeRecordFeatureDetails](#)
- [SzEntityIncludeRecordFeatureStats](#)
- [SzEntityIncludeRecordJsonData](#)
- [SzEntityIncludeRecordUnmappedData](#)

The pre-defined [SzFlag](#) aggregate constants for this group are:

- [SzRecordPreviewDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzRedoFlags = 16

Flags in this usage group can be used for operations that process redo records in the entity repository.

Applicable methods include:

- [ProcessRedoRecord\(string, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzWithInfo](#)

The pre-defined [SzFlag](#) aggregate constants for this group are:

- [SzRedoDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzReevaluateEntityFlags = 8

Flags in this usage group can be used for operations that reevaluate entities in the entity repository.

Applicable methods include:

- [ReevaluateEntity\(long, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzWithInfo](#)

The pre-defined [SzFlag](#) aggregate constants for this group are:

- [SzReevaluateEntityDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzReevaluateRecordFlags = 4

Flags in this usage group can be used for operations that reevaluate records in the entity repository.

Applicable methods include:

- [ReevaluateRecord\(string, string, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzWithInfo](#)

The pre-defined [SzFlag](#) aggregate constants for this group are:

- [SzReevaluateRecordDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzSearchFlags = 2048

Flags in this usage group can be used for operations that search for entities to control how the entities are qualified for inclusion in the search results and the level of detail for the entities returned in the search results.

Applicable methods include:

- [SearchByAttributes\(string, SzFlag?\)](#)
- [SearchByAttributes\(string, string, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzEntityIncludePossiblySameRelations](#)
- [SzEntityIncludePossiblyRelatedRelations](#)
- [SzEntityIncludeNameOnlyRelations](#)

- [SzEntityIncludeDisclosedRelations](#)
- [SzEntityIncludeAllFeatures](#)
- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)
- [SzEntityIncludeRecordTypes](#)
- [SzEntityIncludeRecordData](#)
- [SzEntityIncludeRecordMatchingInfo](#)
- [SzEntityIncludeRecordJsonData](#)
- [SzEntityIncludeRecordUnmappedData](#)
- [SzEntityIncludeRecordFeatures](#)
- [SzEntityIncludeRecordFeatureDetails](#)
- [SzEntityIncludeRecordFeatureStats](#)
- [SzEntityIncludeRelatedEntityName](#)
- [SzEntityIncludeRelatedMatchingInfo](#)
- [SzEntityIncludeRelatedRecordSummary](#)
- [SzEntityIncludeRelatedRecordTypes](#)
- [SzEntityIncludeRelatedRecordData](#)
- [SzEntityIncludeInternalFeatures](#)
- [SzEntityIncludeFeatureStats](#)
- [SzIncludeMatchKeyDetails](#)
- [SzIncludeFeatureScores](#)
- [SzSearchIncludeStats](#)
- [SzSearchIncludeResolved](#)
- [SzSearchIncludePossiblySame](#)
- [SzSearchIncludePossiblyRelated](#)
- [SzSearchIncludeNameOnly](#)
- [SzSearchIncludeAllCandidates](#)
- [SzSearchIncludeRequest](#)
- [SzSearchIncludeRequestDetails](#)

The pre-defined [SzFlag](#) aggregate constants that use this group and are defined for "search" operations are:

- [SzSearchIncludeAllEntities](#)
- [SzSearchByAttributesAll](#)
- [SzSearchByAttributesStrong](#)
- [SzSearchByAttributesMinimalAll](#)
- [SzSearchByAttributesMinimalStrong](#)
- [SzSearchByAttributesDefaultFlags](#)

The pre-defined [SzFlag](#) aggregate constants that also support this group for defining entity or record detail levels are:

- [SzRecordDefaultFlags](#)
- [SzEntityIncludeAllRelations](#)
- [SzEntityDefaultFlags](#)
- [SzEntityBriefDefaultFlags](#)
- [SzVirtualEntityDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzVirtualEntityFlags = 262144

Flags in this usage group can be used for operations that retrieve virtual entities in order to control the level of detail of the returned virtual entity.

Applicable methods include:

- [GetVirtualEntity\(ISet<\(string dataSourceCode, string recordID\)>, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzEntityIncludeAllFeatures](#)
- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)
- [SzEntityIncludeRecordTypes](#)
- [SzEntityIncludeRecordData](#)
- [SzEntityIncludeRecordMatchingInfo](#)
- [SzEntityIncludeRecordJsonData](#)
- [SzEntityIncludeRecordUnmappedData](#)
- [SzEntityIncludeRecordFeatures](#)
- [SzEntityIncludeRecordFeatureDetails](#)
- [SzEntityIncludeRecordFeatureStats](#)
- [SzEntityIncludeInternalFeatures](#)
- [SzEntityIncludeFeatureStats](#)

The pre-defined [SzFlag](#) aggregate constants for this group are:

- [SzVirtualEntityDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzWhyEntitiesFlags = 32768

Flags in this usage group can be used to control the methodology for performing "why analysis", what details to include for the analysis and the level of detail for the entities in the network that are returned.

Applicable methods include:

- [WhyEntities\(long, long, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzEntityIncludePossiblySameRelations](#)
- [SzEntityIncludePossiblyRelatedRelations](#)
- [SzEntityIncludeNameOnlyRelations](#)
- [SzEntityIncludeDisclosedRelations](#)
- [SzEntityIncludeAllFeatures](#)
- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)
- [SzEntityIncludeRecordTypes](#)
- [SzEntityIncludeRecordData](#)
- [SzEntityIncludeRecordMatchingInfo](#)
- [SzEntityIncludeRecordJsonData](#)
- [SzEntityIncludeRecordUnmappedData](#)
- [SzEntityIncludeRecordFeatures](#)
- [SzEntityIncludeRecordFeatureDetails](#)
- [SzEntityIncludeRecordFeatureStats](#)
- [SzEntityIncludeRelatedEntityName](#)
- [SzEntityIncludeRelatedMatchingInfo](#)
- [SzEntityIncludeRelatedRecordSummary](#)
- [SzEntityIncludeRelatedRecordTypes](#)
- [SzEntityIncludeRelatedRecordData](#)
- [SzEntityIncludeInternalFeatures](#)
- [SzEntityIncludeFeatureStats](#)
- [SzIncludeMatchKeyDetails](#)
- [SzIncludeFeatureScores](#)

The pre-defined [SzFlag](#) aggregate constants that use this group and are defined for "why" operations are:

- [SzWhyEntitiesDefaultFlags](#)

The pre-defined [SzFlag](#) aggregate constants that also support this group for defining entity or record detail levels are:

- [SzRecordDefaultFlags](#)
- [SzEntityIncludeAllRelations](#)
- [SzEntityDefaultFlags](#)
- [SzEntityBriefDefaultFlags](#)
- [SzVirtualEntityDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzWhyRecordInEntityFlags = 8192

Flags in this usage group can be used to control the methodology for performing "why analysis", what details to include for the analysis and the level of detail for the entities in the network that are returned.

Applicable methods include:

- [WhyRecordInEntity\(string, string, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzEntityIncludePossiblySameRelations](#)
- [SzEntityIncludePossiblyRelatedRelations](#)
- [SzEntityIncludeNameOnlyRelations](#)
- [SzEntityIncludeDisclosedRelations](#)
- [SzEntityIncludeAllFeatures](#)
- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)
- [SzEntityIncludeRecordTypes](#)
- [SzEntityIncludeRecordData](#)
- [SzEntityIncludeRecordMatchingInfo](#)
- [SzEntityIncludeRecordJsonData](#)
- [SzEntityIncludeRecordUnmappedData](#)
- [SzEntityIncludeRecordFeatures](#)
- [SzEntityIncludeRecordFeatureDetails](#)
- [SzEntityIncludeRecordFeatureStats](#)
- [SzEntityIncludeRelatedEntityName](#)
- [SzEntityIncludeRelatedMatchingInfo](#)
- [SzEntityIncludeRelatedRecordSummary](#)
- [SzEntityIncludeRelatedRecordTypes](#)
- [SzEntityIncludeRelatedRecordData](#)
- [SzEntityIncludeInternalFeatures](#)
- [SzEntityIncludeFeatureStats](#)

- [SzIncludeMatchKeyDetails](#)
- [SzIncludeFeatureScores](#)

The pre-defined [SzFlag](#) aggregate constants that use this group and are defined for "why" operations are:

- [SzWhyRecordInEntityDefaultFlags](#)

The pre-defined [SzFlag](#) aggregate constants that also support this group for defining entity or record detail levels are:

- [SzRecordDefaultFlags](#)
- [SzEntityIncludeAllRelations](#)
- [SzEntityDefaultFlags](#)
- [SzEntityBriefDefaultFlags](#)
- [SzVirtualEntityDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzWhyRecordsFlags = 16384

Flags in this usage group can be used to control the methodology for performing "why analysis", what details to include for the analysis and the level of detail for the entities in the network that are returned.

Applicable methods include:

- [WhyRecords\(string, string, string, string, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzEntityIncludePossiblySameRelations](#)
- [SzEntityIncludePossiblyRelatedRelations](#)
- [SzEntityIncludeNameOnlyRelations](#)
- [SzEntityIncludeDisclosedRelations](#)
- [SzEntityIncludeAllFeatures](#)
- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)
- [SzEntityIncludeRecordTypes](#)
- [SzEntityIncludeRecordData](#)
- [SzEntityIncludeRecordMatchingInfo](#)
- [SzEntityIncludeRecordJsonData](#)
- [SzEntityIncludeRecordUnmappedData](#)

- [SzEntityIncludeRecordFeatures](#)
- [SzEntityIncludeRecordFeatureDetails](#)
- [SzEntityIncludeRecordFeatureStats](#)
- [SzEntityIncludeRelatedEntityName](#)
- [SzEntityIncludeRelatedMatchingInfo](#)
- [SzEntityIncludeRelatedRecordSummary](#)
- [SzEntityIncludeRelatedRecordTypes](#)
- [SzEntityIncludeRelatedRecordData](#)
- [SzEntityIncludeInternalFeatures](#)
- [SzEntityIncludeFeatureStats](#)
- [SzIncludeMatchKeyDetails](#)
- [SzIncludeFeatureScores](#)

The pre-defined [SzFlag](#) aggregate constants that use this group and are defined for "why" operations are:

- [SzWhyRecordsDefaultFlags](#)

The pre-defined [SzFlag](#) aggregate constants that also support this group for defining entity or record detail levels are:

- [SzRecordDefaultFlags](#)
- [SzEntityIncludeAllRelations](#)
- [SzEntityDefaultFlags](#)
- [SzEntityBriefDefaultFlags](#)
- [SzVirtualEntityDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html>

SzWhySearchFlags = 65536

Flags in this usage group can be used for operations that search for entities to control how the entities are qualified for inclusion in the search results and the level of detail for the entities returned in the search results.

Applicable methods include:

- [WhySearch\(string, long, string, SzFlag?\)](#)

The [SzFlag](#) instances included in this usage group are:

- [SzEntityIncludePossiblySameRelations](#)
- [SzEntityIncludePossiblyRelatedRelations](#)
- [SzEntityIncludeNameOnlyRelations](#)

- [SzEntityIncludeDisclosedRelations](#)
- [SzEntityIncludeAllFeatures](#)
- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)
- [SzEntityIncludeRecordTypes](#)
- [SzEntityIncludeRecordData](#)
- [SzEntityIncludeRecordMatchingInfo](#)
- [SzEntityIncludeRecordJsonData](#)
- [SzEntityIncludeRecordUnmappedData](#)
- [SzEntityIncludeRecordFeatures](#)
- [SzEntityIncludeRecordFeatureDetails](#)
- [SzEntityIncludeRecordFeatureStats](#)
- [SzEntityIncludeRelatedEntityName](#)
- [SzEntityIncludeRelatedMatchingInfo](#)
- [SzEntityIncludeRelatedRecordSummary](#)
- [SzEntityIncludeRelatedRecordTypes](#)
- [SzEntityIncludeRelatedRecordData](#)
- [SzEntityIncludeInternalFeatures](#)
- [SzEntityIncludeFeatureStats](#)
- [SzIncludeMatchKeyDetails](#)
- [SzIncludeFeatureScores](#)
- [SzSearchIncludeStats](#)
- [SzSearchIncludeRequest](#)
- [SzSearchIncludeRequestDetails](#)

The pre-defined [SzFlag](#) aggregate constants that use this group and are defined for "search" operations are:

- [SzWhySearchDefaultFlags](#)

The pre-defined [SzFlag](#) aggregate constants that also support this group for defining entity or record detail levels are:

- [SzRecordDefaultFlags](#)
- [SzEntityIncludeAllRelations](#)
- [SzEntityDefaultFlags](#)
- [SzEntityBriefDefaultFlags](#)
- [SzVirtualEntityDefaultFlags](#)

See also: <https://docs.senzing.com/flags/index.html> ↗

Class SzFlags

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Provides aggregate [SzFlag](#) constants as well as extension methods and utility methods pertaining to [SzFlag](#) and [SzFlagUsageGroup](#).

```
public static class SzFlags
```

Inheritance

[object](#) ← SzFlags

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Fields

SzAddRecordAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzAddRecordFlags](#) usage group.

```
public const SzFlag SzAddRecordAllFlags = SzWithInfo
```

Field Value

[SzFlag](#)

See Also

[SzAddRecordFlags](#)

SzAddRecordDefaultFlags

The aggregate [SzFlag](#) representing the default flags for "add record" operations.

```
public const SzFlag SzAddRecordDefaultFlags = (SzFlag)0
```

Field Value

[SzFlag](#)

Remarks

Currently, this is equivalent to [SzNoFlags](#).

All the flags in this constant belong to the following usage groups:

- [SzAddRecordFlags](#)

See Also

<https://docs.senzing.com/flags/index.html> ↗

SzDeleteRecordAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzDeleteRecordFlags](#) usage group.

```
public const SzFlag SzDeleteRecordAllFlags = SzWithInfo
```

Field Value

[SzFlag](#)

See Also

[SzDeleteRecordFlags](#)

SzDeleteRecordDefaultFlags

The aggregate [SzFlag](#) representing the default flags for "delete record" operations.

```
public const SzFlag SzDeleteRecordDefaultFlags = (SzFlag)0
```

Field Value

[SzFlag](#)

Remarks

Currently, this is equivalent to [SzNoFlags](#).

All the flags in this constant belong to the following usage groups:

- [SzDeleteRecordFlags](#)

See Also

<https://docs.senzing.com/flags/index.html> ↗

SzEntityAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzEntityFlags](#) usage group.

```
public const SzFlag SzEntityAllFlags = SzEntityIncludePossiblySameRelations |  
SzEntityIncludePossiblyRelatedRelations | SzEntityIncludeNameOnlyRelations |  
SzEntityIncludeDisclosedRelations | SzEntityIncludeAllFeatures |  
SzEntityIncludeRepresentativeFeatures | SzEntityIncludeEntityName |  
SzEntityIncludeRecordSummary | SzEntityIncludeRecordData |  
SzEntityIncludeRecordMatchingInfo | SzEntityIncludeRecordJsonData |  
SzEntityIncludeRecordFeatures | SzEntityIncludeRelatedEntityName |  
SzEntityIncludeRelatedMatchingInfo | SzEntityIncludeRelatedRecordSummary |  
SzEntityIncludeRelatedRecordData | SzEntityIncludeInternalFeatures |  
SzEntityIncludeFeatureStats | SzEntityIncludeRecordTypes |  
SzEntityIncludeRelatedRecordTypes | SzEntityIncludeRecordUnmappedData |  
SzIncludeMatchKeyDetails | SzEntityIncludeRecordFeatureDetails |  
SzEntityIncludeRecordFeatureStats | SzEntityIncludeRecordDates
```

Field Value

[SzFlag](#)

See Also

[SzEntityFlags](#)

SzEntityBriefDefaultFlags

The aggregate [SzFlag](#) to produce the default level of detail when retrieving entities.

```
public const SzFlag SzEntityBriefDefaultFlags = SzEntityIncludePossiblySameRelations  
| SzEntityIncludePossiblyRelatedRelations | SzEntityIncludeNameOnlyRelations |  
SzEntityIncludeDisclosedRelations | SzEntityIncludeRecordMatchingInfo |  
SzEntityIncludeRelatedMatchingInfo
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- All flags from [SzEntityIncludeAllRelations](#)
- [SzEntityIncludeRecordMatchingInfo](#)
- [SzEntityIncludeRelatedMatchingInfo](#)

All the flags in this constant belong to the [SzEntityFlags](#) usage group, and by extension belong to the following usage groups which are super sets:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzEntityCoreFlags

The aggregate [SzFlag](#) to produce basic entity output without any related entity data.

```
public const SzFlag SzEntityCoreFlags = SzEntityIncludeRepresentativeFeatures |  
SzEntityIncludeEntityName | SzEntityIncludeRecordSummary | SzEntityIncludeRecordData  
| SzEntityIncludeRecordMatchingInfo
```

Field Value

[SzFlag](#)

Remarks

This constant may be used directly but is primarily used as a basis for other aggregate constants.

The included [SzFlag](#) values are:

- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)
- [SzEntityIncludeRecordData](#)
- [SzEntityIncludeRecordMatchingInfo](#)

All the flags in this constant belong to the [SzEntityFlags](#) usage group, and by extension belong to the following usage groups which are super sets:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See Also

<https://docs.senzing.com/flags/index.html> ↗

SzEntityDefaultFlags

The aggregate [SzFlag](#) to produce the default level of detail when retrieving entities.

```
public const SzFlag SzEntityDefaultFlags = SzEntityIncludePossiblySameRelations |  
SzEntityIncludePossiblyRelatedRelations | SzEntityIncludeNameOnlyRelations |  
SzEntityIncludeDisclosedRelations | SzEntityIncludeRepresentativeFeatures |  
SzEntityIncludeEntityName | SzEntityIncludeRecordSummary | SzEntityIncludeRecordData  
| SzEntityIncludeRecordMatchingInfo | SzEntityIncludeRelatedEntityName |  
SzEntityIncludeRelatedMatchingInfo | SzEntityIncludeRelatedRecordSummary
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- All flags from [SzEntityCoreFlags](#)
- All flags from [SzEntityIncludeAllRelations](#)
- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeRelatedEntityName](#)
- [SzEntityIncludeRelatedRecordSummary](#)
- [SzEntityIncludeRelatedMatchingInfo](#)

All the flags in this constant belong to the [SzEntityFlags](#) usage group, and by extension belong to the following usage groups which are super sets:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See Also

<https://docs.senzing.com/flags/index.html> ↗

SzEntityIncludeAllRelations

The aggregate [SzFlag](#) for including all relations for entities.

```
public const SzFlag SzEntityIncludeAllRelations =  
SzEntityIncludePossiblySameRelations | SzEntityIncludePossiblyRelatedRelations |  
SzEntityIncludeNameOnlyRelations | SzEntityIncludeDisclosedRelations
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzEntityIncludePossiblySameRelations](#)
- [SzEntityIncludePossiblyRelatedRelations](#)
- [SzEntityIncludeNameOnlyRelations](#)
- [SzEntityIncludeDisclosedRelations](#)

All the flags in this constant belong to the [SzEntityFlags](#) usage group, and by extension belong to the following usage groups which are super sets:

- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzExportAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzExportFlags](#) usage group.

```
public const SzFlag SzExportAllFlags = SzExportIncludeMultiRecordEntities |  
SzExportIncludePossiblySame | SzExportIncludePossiblyRelated |  
SzExportIncludeNameOnly | SzExportIncludeDisclosed |  
SzExportIncludeSingleRecordEntities | SzEntityIncludePossiblySameRelations |  
SzEntityIncludePossiblyRelatedRelations | SzEntityIncludeNameOnlyRelations |  
SzEntityIncludeDisclosedRelations | SzEntityIncludeAllFeatures |  
SzEntityIncludeRepresentativeFeatures | SzEntityIncludeEntityName |  
SzEntityIncludeRecordSummary | SzEntityIncludeRecordData |  
SzEntityIncludeRecordMatchingInfo | SzEntityIncludeRecordJsonData |  
SzEntityIncludeRecordFeatures | SzEntityIncludeRelatedEntityName |  
SzEntityIncludeRelatedMatchingInfo | SzEntityIncludeRelatedRecordSummary |  
SzEntityIncludeRelatedRecordData | SzEntityIncludeInternalFeatures |  
SzEntityIncludeFeatureStats | SzEntityIncludeRecordTypes |  
SzEntityIncludeRelatedRecordTypes | SzEntityIncludeRecordUnmappedData |
```

```
SzIncludeMatchKeyDetails | SzEntityIncludeRecordFeatureDetails |  
SzEntityIncludeRecordFeatureStats | SzEntityIncludeRecordDates
```

Field Value

[SzFlag](#)

See Also

[SzExportFlags](#)

SzExportDefaultFlags

The aggregate [SzFlag](#) representing the default settings for exporting entities.

```
public const SzFlag SzExportDefaultFlags = SzExportIncludeMultiRecordEntities |  
SzExportIncludeSingleRecordEntities | SzEntityIncludePossiblySameRelations |  
SzEntityIncludePossiblyRelatedRelations | SzEntityIncludeNameOnlyRelations |  
SzEntityIncludeDisclosedRelations | SzEntityIncludeRepresentativeFeatures |  
SzEntityIncludeEntityName | SzEntityIncludeRecordSummary | SzEntityIncludeRecordData  
| SzEntityIncludeRecordMatchingInfo | SzEntityIncludeRelatedEntityName |  
SzEntityIncludeRelatedMatchingInfo | SzEntityIncludeRelatedRecordSummary
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- All flags from [SzExportIncludeAllEntities](#)
- All flags from [SzEntityDefaultFlags](#)

All the flags in this constant belong to the following usage groups:

- [SzExportFlags](#)

See Also

<https://docs.senzing.com/flags/index.html> ↗

SzExportIncludeAllEntities

The aggregate [SzFlag](#) for export functionality to indicate that all entities should be included in an export.

```
public const SzFlag SzExportIncludeAllEntities = SzExportIncludeMultiRecordEntities  
| SzExportIncludeSingleRecordEntities
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzExportIncludeMultiRecordEntities](#)
- [SzExportIncludeSingleRecordEntities](#)

All the flags in this constant belong to the following usage groups:

- [SzExportFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzExportIncludeAllHavingRelationships

The aggregate [SzFlag](#) for export functionality to indicate that entities having relationships of any kind should be included in an export.

```
public const SzFlag SzExportIncludeAllHavingRelationships =  
SzExportIncludePossiblySame | SzExportIncludePossiblyRelated |  
SzExportIncludeNameOnly | SzExportIncludeDisclosed
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzExportIncludePossiblySame](#)
- [SzExportIncludePossiblyRelated](#)
- [SzExportIncludeNameOnly](#)
- [SzExportIncludeDisclosed](#)

All the flags in this constant belong to the following usage groups:

- [SzExportFlags](#)

See Also

<https://docs.senzing.com/flags/index.html> ↗

SzFindInterestingEntitiesAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzFindInterestingEntitiesFlags](#) usage group.

```
public const SzFlag SzFindInterestingEntitiesAllFlags = (SzFlag)0
```

Field Value

[SzFlag](#)

See Also

[SzFindInterestingEntitiesFlags](#)

SzFindInterestingEntitiesDefaultFlags

The aggregate [SzFlag](#) representing the default flags for "find interesting entities" operations.

```
public const SzFlag SzFindInterestingEntitiesDefaultFlags = (SzFlag)0
```

Field Value

[SzFlag](#)

Remarks

Currently, this is equivalent to [SzNoFlags](#).

All the flags in this constant belong to the following usage groups:

- [SzFindInterestingEntitiesFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzFindNetworkAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzFindNetworkFlags](#) usage group.

```
public const SzFlag SzFindNetworkAllFlags = SzEntityIncludePossiblySameRelations |  
SzEntityIncludePossiblyRelatedRelations | SzEntityIncludeNameOnlyRelations |  
SzEntityIncludeDisclosedRelations | SzEntityIncludeAllFeatures |  
SzEntityIncludeRepresentativeFeatures | SzEntityIncludeEntityName |  
SzEntityIncludeRecordSummary | SzEntityIncludeRecordData |  
SzEntityIncludeRecordMatchingInfo | SzEntityIncludeRecordJsonData |  
SzEntityIncludeRecordFeatures | SzEntityIncludeRelatedEntityName |  
SzEntityIncludeRelatedMatchingInfo | SzEntityIncludeRelatedRecordSummary |  
SzEntityIncludeRelatedRecordData | SzEntityIncludeInternalFeatures |  
SzEntityIncludeFeatureStats | SzEntityIncludeRecordTypes |  
SzEntityIncludeRelatedRecordTypes | SzEntityIncludeRecordUnmappedData  
| SzFindNetworkIncludeMatchingInfo | SzIncludeMatchKeyDetails |  
SzEntityIncludeRecordFeatureDetails | SzEntityIncludeRecordFeatureStats  
| SzEntityIncludeRecordDates
```

Field Value

[SzFlag](#)

See Also

[SzFindNetworkFlags](#)

SzFindNetworkDefaultFlags

The aggregate [SzFlag](#) representing the default settings for finding entity networks.

```
public const SzFlag SzFindNetworkDefaultFlags = SzEntityIncludeEntityName |  
SzEntityIncludeRecordSummary | SzFindNetworkIncludeMatchingInfo
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzFindNetworkIncludeMatchingInfo](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)

All the flags in this constant belong to the following usage groups:

- [SzFindNetworkFlags](#)

See Also

<https://docs.senzing.com/flags/index.html> ↗

SzFindPathAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzFindPathFlags](#) usage group.

```
public const SzFlag SzFindPathAllFlags = SzEntityIncludePossiblySameRelations |  
SzEntityIncludePossiblyRelatedRelations | SzEntityIncludeNameOnlyRelations |  
SzEntityIncludeDisclosedRelations | SzEntityIncludeAllFeatures |  
SzEntityIncludeRepresentativeFeatures | SzEntityIncludeEntityName |  
SzEntityIncludeRecordSummary | SzEntityIncludeRecordData |  
SzEntityIncludeRecordMatchingInfo | SzEntityIncludeRecordJsonData |  
SzEntityIncludeRecordFeatures | SzEntityIncludeRelatedEntityName |  
SzEntityIncludeRelatedMatchingInfo | SzEntityIncludeRelatedRecordSummary |  
SzEntityIncludeRelatedRecordData | SzEntityIncludeInternalFeatures |  
SzEntityIncludeFeatureStats | SzFindPathStrictAvoid | SzEntityIncludeRecordTypes |  
SzEntityIncludeRelatedRecordTypes | SzFindPathIncludeMatchingInfo  
| SzEntityIncludeRecordUnmappedData | SzIncludeMatchKeyDetails |  
SzEntityIncludeRecordFeatureDetails | SzEntityIncludeRecordFeatureStats  
| SzEntityIncludeRecordDates
```

Field Value

[SzFlag](#)

See Also

[SzFindPathFlags](#)

SzFindPathDefaultFlags

The aggregate [SzFlag](#) representing the default settings for finding entity paths.

```
public const SzFlag SzFindPathDefaultFlags = SzEntityIncludeEntityName |  
SzEntityIncludeRecordSummary | SzFindPathIncludeMatchingInfo
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzFindPathIncludeMatchingInfo](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)

All the flags in this constant belong to the following usage groups:

- [SzFindPathFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzHowAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzHowFlags](#) usage group.

```
public const SzFlag SzHowAllFlags = SzIncludeFeatureScores  
| SzIncludeMatchKeyDetails
```

Field Value

[SzFlag](#).

See Also

[SzHowFlags](#)

SzHowEntityDefaultFlags

The aggregate [SzFlag](#) representing the default settings for "how entity" operations.

```
public const SzFlag SzHowEntityDefaultFlags = SzIncludeFeatureScores
```

Field Value

[SzFlag](#).

Remarks

The included [SzFlag](#) values are:

- [SzIncludeFeatureScores](#)

All the flags in this constant belong to the following usage groups:

- [SzHowFlags](#)

See Also

<https://docs.senzing.com/flags/index.html> ↗

SzNoFlagUsageGroups

The value representing no [SzFlagUsageGroup](#) values (i.e.: zero typed as the enum).

```
public const SzFlagUsageGroup SzNoFlagUsageGroups = (SzFlagUsageGroup)0
```

Field Value

[SzFlagUsageGroup](#)

SzNoFlags

The value representing no flags are being passed.

```
public const SzFlag SzNoFlags = (SzFlag)0
```

Field Value

[SzFlag](#)

Remarks

Alternatively, a `null` value will indicate no flags as well.

SzRecordAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzRecordFlags](#) usage group.

```
public const SzFlag SzRecordAllFlags = SzEntityIncludeRecordJsonData |  
SzEntityIncludeRecordFeatures | SzEntityIncludeInternalFeatures |  
SzEntityIncludeRecordUnmappedData | SzEntityIncludeRecordFeatureDetails |  
SzEntityIncludeRecordFeatureStats | SzEntityIncludeRecordDates
```

Field Value

[SzFlag](#)

See Also

[SzRecordFlags](#)

SzRecordDefaultFlags

The aggregate [SzFlag](#) to produce the default level of detail when retrieving records.

```
public const SzFlag SzRecordDefaultFlags = SzEntityIncludeRecordJsonData
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzEntityIncludeRecordJsonData](#)

All the flags in this constant belong to the [SzRecordFlags](#) usage group, and by extension belong to the following usage groups which are super sets:

- [SzRecordFlags](#)
- [SzEntityFlags](#)
- [SzSearchFlags](#)
- [SzExportFlags](#)
- [SzFindPathFlags](#)
- [SzFindNetworkFlags](#)
- [SzWhyRecordInEntityFlags](#)
- [SzWhyRecordsFlags](#)
- [SzWhyEntitiesFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzRecordPreviewAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzRecordPreviewFlags](#) usage group.

```
public const SzFlag SzRecordPreviewAllFlags = SzEntityIncludeRecordJsonData |  
SzEntityIncludeRecordFeatures | SzEntityIncludeInternalFeatures |  
SzEntityIncludeRecordUnmappedData | SzEntityIncludeRecordFeatureDetails |  
SzEntityIncludeRecordFeatureStats
```

Field Value

[SzFlag](#)

See Also

[SzRecordPreviewFlags](#)

SzRecordPreviewDefaultFlags

The aggregate [SzFlag](#) representing the default flags for [record preview](#) operations.

```
public const SzFlag SzRecordPreviewDefaultFlags =  
SzEntityIncludeRecordFeatureDetails
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzEntityIncludeRecordFeatureDetails](#)

All the flags in this constant belong to the following usage groups:

- [SzRecordPreviewFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzRedoAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzRedoFlags](#) usage group.

```
public const SzFlag SzRedoAllFlags = SzWithInfo
```

Field Value

[SzFlag](#)

See Also

[SzDeleteRecordFlags](#)

SzRedoDefaultFlags

The aggregate [SzFlag](#) representing the default flags for "process redo" operations.

```
public const SzFlag SzRedoDefaultFlags = (SzFlag)0
```

Field Value

[SzFlag](#)

Remarks

Currently, this is equivalent to [SzNoFlags](#).

All the flags in this constant belong to the following usage groups:

- [SzRedoFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzReevaluateEntityAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzReevaluateEntityFlags](#) usage group.

```
public const SzFlag SzReevaluateEntityAllFlags = SzWithInfo
```

Field Value

[SzFlag](#)

See Also

[SzReevaluateEntityFlags](#)

SzReevaluateEntityDefaultFlags

The aggregate [SzFlag](#) representing the default flags for "reevaluate entity" operations.

```
public const SzFlag SzReevaluateEntityDefaultFlags = (SzFlag)0
```

Field Value

[SzFlag](#)

Remarks

Currently, this is equivalent to [SzNoFlags](#).

All the flags in this constant belong to the following usage groups:

- [SzReevaluateEntityFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzReevaluateRecordAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzReevaluateRecordFlags](#) usage group.

```
public const SzFlag SzReevaluateRecordAllFlags = SzWithInfo
```

Field Value

[SzFlag](#)

See Also

[SzReevaluateRecordFlags](#)

SzReevaluateRecordDefaultFlags

The aggregate [SzFlag](#) representing the default flags for "reevaluate record" operations.

```
public const SzFlag SzReevaluateRecordDefaultFlags = (SzFlag)0
```

Field Value

[SzFlag](#)

Remarks

Currently, this is equivalent to [SzNoFlags](#).

All the flags in this constant belong to the following usage groups:

- [SzReevaluateRecordFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzSearchAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzSearchFlags](#) usage group.

```
public const SzFlag SzSearchAllFlags = SzExportIncludeMultiRecordEntities |  
SzExportIncludePossiblySame | SzExportIncludePossiblyRelated |  
SzExportIncludeNameOnly | SzEntityIncludePossiblySameRelations |  
SzEntityIncludePossiblyRelatedRelations | SzEntityIncludeNameOnlyRelations |  
SzEntityIncludeDisclosedRelations | SzEntityIncludeAllFeatures |  
SzEntityIncludeRepresentativeFeatures | SzEntityIncludeEntityName |  
SzEntityIncludeRecordSummary | SzEntityIncludeRecordData |  
SzEntityIncludeRecordMatchingInfo | SzEntityIncludeRecordJsonData |  
SzEntityIncludeRecordFeatures | SzEntityIncludeRelatedEntityName |  
SzEntityIncludeRelatedMatchingInfo | SzEntityIncludeRelatedRecordSummary |  
SzEntityIncludeRelatedRecordData | SzEntityIncludeInternalFeatures |  
SzEntityIncludeFeatureStats | SzIncludeFeatureScores | SzSearchIncludeStats |  
SzEntityIncludeRecordTypes | SzEntityIncludeRelatedRecordTypes |  
SzEntityIncludeRecordUnmappedData | SzSearchIncludeAllCandidates |  
SzIncludeMatchKeyDetails | SzEntityIncludeRecordFeatureDetails |  
SzEntityIncludeRecordFeatureStats | SzSearchIncludeRequest |  
SzSearchIncludeRequestDetails | SzEntityIncludeRecordDates
```

Field Value

[SzFlag](#)

See Also

[SzSearchFlags](#)

SzSearchByAttributesAll

The aggregate [SzFlag](#) indicating that search results should include all matching entities regardless of match level.

```
public const SzFlag SzSearchByAttributesAll = SzExportIncludeMultiRecordEntities |  
SzExportIncludePossiblySame | SzExportIncludePossiblyRelated |  
SzExportIncludeNameOnly | SzEntityIncludeRepresentativeFeatures |  
SzEntityIncludeEntityName | SzEntityIncludeRecordSummary | SzIncludeFeatureScores  
| SzSearchIncludeStats
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- All flags from [SzSearchIncludeAllEntities](#)
- [SzSearchIncludeStats](#)
- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)
- [SzIncludeFeatureScores](#)

All the flags in this constant belong to the following usage groups:

- [SzSearchFlags](#)

See Also

<https://docs.senzing.com/flags/index.html> ↗

SzSearchByAttributesDefaultFlags

The aggregate [SzFlag](#) representing the default flags for "search by attribute" operations.

```
public const SzFlag SzSearchByAttributesDefaultFlags =  
SzExportIncludeMultiRecordEntities | SzExportIncludePossiblySame |  
SzExportIncludePossiblyRelated | SzExportIncludeNameOnly |
```

```
SzEntityIncludeRepresentativeFeatures | SzEntityIncludeEntityName |
SzEntityIncludeRecordSummary | SzIncludeFeatureScores | SzSearchIncludeStats
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- All flags from [SzSearchByAttributesAll](#)

All the flags in this constant belong to the following usage groups:

- [SzSearchFlags](#)

See Also

<https://docs.senzing.com/flags/index.html> ↗

SzSearchByAttributesMinimalAll

The aggregate [SzFlag](#) indicating that search results should include all matching entities regardless of match level while returning minimal data for those entities.

```
public const SzFlag SzSearchByAttributesMinimalAll =
SzExportIncludeMultiRecordEntities | SzExportIncludePossiblySame |
SzExportIncludePossiblyRelated | SzExportIncludeNameOnly | SzSearchIncludeStats
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- All flags from [SzSearchIncludeAllEntities](#)
- [SzSearchIncludeStats](#)

All the flags in this constant belong to the following usage groups:

- [SzSearchFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzSearchByAttributesMinimalStrong

The aggregate [SzFlag](#) indicating that search results should only include strongly matching entities while returning minimal data for those matching entities.

```
public const SzFlag SzSearchByAttributesMinimalStrong =  
SzExportIncludeMultiRecordEntities | SzExportIncludePossiblySame  
| SzSearchIncludeStats
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzSearchIncludeResolved](#)
- [SzSearchIncludePossiblySame](#)
- [SzSearchIncludeStats](#)

All the flags in this constant belong to the following usage groups:

- [SzSearchFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzSearchByAttributesStrong

The aggregate [SzFlag](#) indicating that search results should only include strongly matching entities.

```
public const SzFlag SzSearchByAttributesStrong = SzExportIncludeMultiRecordEntities  
| SzExportIncludePossiblySame | SzEntityIncludeRepresentativeFeatures |
```

```
SzEntityIncludeEntityName | SzEntityIncludeRecordSummary | SzIncludeFeatureScores  
| SzSearchIncludeStats
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzSearchIncludeResolved](#)
- [SzSearchIncludePossiblySame](#)
- [SzSearchIncludeStats](#)
- [SzEntityIncludeRepresentativeFeatures](#)
- [SzEntityIncludeEntityName](#)
- [SzEntityIncludeRecordSummary](#)
- [SzIncludeFeatureScores](#)

All the flags in this constant belong to the following usage groups:

- [SzSearchFlags](#)

See Also

<https://docs.senzing.com/flags/index.html> ↗

SzSearchIncludeAllEntities

The aggregate [SzFlag](#) for search functionality to indicate that search results should include entities that are related to the search attributes at any match level.

```
public const SzFlag SzSearchIncludeAllEntities = SzExportIncludeMultiRecordEntities  
| SzExportIncludePossiblySame | SzExportIncludePossiblyRelated  
| SzExportIncludeNameOnly
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzSearchIncludeResolved](#)
- [SzSearchIncludePossiblySame](#)
- [SzSearchIncludePossiblyRelated](#)
- [SzSearchIncludeNameOnly](#)

All the flags in this constant belong to the following usage groups:

- [SzSearchFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzVirtualEntityAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzVirtualEntityFlags](#) usage group.

```
public const SzFlag SzVirtualEntityAllFlags = SzEntityIncludeAllFeatures |  
SzEntityIncludeRepresentativeFeatures | SzEntityIncludeEntityName |  
SzEntityIncludeRecordSummary | SzEntityIncludeRecordData |  
SzEntityIncludeRecordMatchingInfo | SzEntityIncludeRecordJsonData |  
SzEntityIncludeRecordFeatures | SzEntityIncludeInternalFeatures |  
SzEntityIncludeFeatureStats | SzEntityIncludeRecordTypes |  
SzEntityIncludeRecordUnmappedData | SzEntityIncludeRecordFeatureDetails |  
SzEntityIncludeRecordFeatureStats | SzEntityIncludeRecordDates
```

Field Value

[SzFlag](#)

See Also

[SzVirtualEntityFlags](#)

SzVirtualEntityDefaultFlags

The aggregate [SzFlag](#) representing the default settings when retrieving virtual entities.

```
public const SzFlag SzVirtualEntityDefaultFlags =  
SzEntityIncludeRepresentativeFeatures | SzEntityIncludeEntityName |
```

SzEntityIncludeRecordSummary | SzEntityIncludeRecordData |
SzEntityIncludeRecordMatchingInfo

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- All flags from [SzEntityCoreFlags](#)

All the flags in this constant belong to the following usage groups:

- [SzVirtualEntityFlags](#)

See Also

<https://docs.senzing.com/flags/index.html> ↗

SzWhyEntitiesAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzWhyEntitiesFlags](#) usage group.

```
public const SzFlag SzWhyEntitiesAllFlags = SzEntityIncludePossiblySameRelations |
SzEntityIncludePossiblyRelatedRelations | SzEntityIncludeNameOnlyRelations |
SzEntityIncludeDisclosedRelations | SzEntityIncludeAllFeatures |
SzEntityIncludeRepresentativeFeatures | SzEntityIncludeEntityName |
SzEntityIncludeRecordSummary | SzEntityIncludeRecordData |
SzEntityIncludeRecordMatchingInfo | SzEntityIncludeRecordJsonData |
SzEntityIncludeRecordFeatures | SzEntityIncludeRelatedEntityName |
SzEntityIncludeRelatedMatchingInfo | SzEntityIncludeRelatedRecordSummary |
SzEntityIncludeRelatedRecordData | SzEntityIncludeInternalFeatures |
SzEntityIncludeFeatureStats | SzIncludeFeatureScores | SzEntityIncludeRecordTypes |
SzEntityIncludeRelatedRecordTypes | SzEntityIncludeRecordUnmappedData |
SzIncludeMatchKeyDetails | SzEntityIncludeRecordFeatureDetails |
SzEntityIncludeRecordFeatureStats | SzEntityIncludeRecordDates
```

Field Value

[SzFlag](#)

See Also

[SzWhyRecordInEntityFlags](#)

SzWhyEntitiesDefaultFlags

The aggregate [SzFlag](#) representing the default settings for "why entity" operations.

```
public const SzFlag SzWhyEntitiesDefaultFlags = SzIncludeFeatureScores
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzIncludeFeatureScores](#)

All the flags in this constant belong to the following usage groups:

- [SzWhyEntitiesFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

SzWhyRecordInEntityAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzWhyRecordInEntityFlags](#) usage group.

```
public const SzFlag SzWhyRecordInEntityAllFlags =
SzEntityIncludePossiblySameRelations | SzEntityIncludePossiblyRelatedRelations |
SzEntityIncludeNameOnlyRelations | SzEntityIncludeDisclosedRelations |
SzEntityIncludeAllFeatures | SzEntityIncludeRepresentativeFeatures |
SzEntityIncludeEntityName | SzEntityIncludeRecordSummary | SzEntityIncludeRecordData |
SzEntityIncludeRecordMatchingInfo | SzEntityIncludeRecordJsonData |
SzEntityIncludeRecordFeatures | SzEntityIncludeRelatedEntityName |
SzEntityIncludeRelatedMatchingInfo | SzEntityIncludeRelatedRecordSummary |
SzEntityIncludeRelatedRecordData | SzEntityIncludeInternalFeatures |
```

```
SzEntityIncludeFeatureStats | SzIncludeFeatureScores | SzEntityIncludeRecordTypes |  
SzEntityIncludeRelatedRecordTypes | SzEntityIncludeRecordUnmappedData |  
SzIncludeMatchKeyDetails | SzEntityIncludeRecordFeatureDetails |  
SzEntityIncludeRecordFeatureStats | SzEntityIncludeRecordDates
```

Field Value

[SzFlag](#)

See Also

[SzWhyRecordInEntityFlags](#)

SzWhyRecordInEntityDefaultFlags

The aggregate [SzFlag](#) representing the default settings for "why record in entity" operations.

```
public const SzFlag SzWhyRecordInEntityDefaultFlags = SzIncludeFeatureScores
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzIncludeFeatureScores](#)

All the flags in this constant belong to the following usage groups:

- [SzWhyRecordInEntityFlags](#)

See Also

<https://docs.senzing.com/flags/index.html> ↗

SzWhyRecordsAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzWhyRecordsFlags](#) usage group.

```
public const SzFlag SzWhyRecordsAllFlags = SzEntityIncludePossiblySameRelations |  
SzEntityIncludePossiblyRelatedRelations | SzEntityIncludeNameOnlyRelations |  
SzEntityIncludeDisclosedRelations | SzEntityIncludeAllFeatures |  
SzEntityIncludeRepresentativeFeatures | SzEntityIncludeEntityName |  
SzEntityIncludeRecordSummary | SzEntityIncludeRecordData |  
SzEntityIncludeRecordMatchingInfo | SzEntityIncludeRecordJsonData |  
SzEntityIncludeRecordFeatures | SzEntityIncludeRelatedEntityName |  
SzEntityIncludeRelatedMatchingInfo | SzEntityIncludeRelatedRecordSummary |  
SzEntityIncludeRelatedRecordData | SzEntityIncludeInternalFeatures |  
SzEntityIncludeFeatureStats | SzIncludeFeatureScores | SzEntityIncludeRecordTypes |  
SzEntityIncludeRelatedRecordTypes | SzEntityIncludeRecordUnmappedData |  
SzIncludeMatchKeyDetails | SzEntityIncludeRecordFeatureDetails |  
SzEntityIncludeRecordFeatureStats | SzEntityIncludeRecordDates
```

Field Value

[SzFlag](#)

See Also

[SzWhyRecordInEntityFlags](#)

SzWhyRecordsDefaultFlags

The aggregate [SzFlag](#) representing the default settings for "why records" operations.

```
public const SzFlag SzWhyRecordsDefaultFlags = SzIncludeFeatureScores
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzIncludeFeatureScores](#)

All the flags in this constant belong to the following usage groups:

- [SzWhyRecordsFlags](#)

See Also

SzWhySearchAllFlags

The [SzFlag](#) value that aggregates all [SzFlag](#) constants belonging to the [SzWhySearchFlags](#) usage group.

```
public const SzFlag SzWhySearchAllFlags = SzEntityIncludePossiblySameRelations |  
SzEntityIncludePossiblyRelatedRelations | SzEntityIncludeNameOnlyRelations |  
SzEntityIncludeDisclosedRelations | SzEntityIncludeAllFeatures |  
SzEntityIncludeRepresentativeFeatures | SzEntityIncludeEntityName |  
SzEntityIncludeRecordSummary | SzEntityIncludeRecordData |  
SzEntityIncludeRecordMatchingInfo | SzEntityIncludeRecordJsonData |  
SzEntityIncludeRecordFeatures | SzEntityIncludeRelatedEntityName |  
SzEntityIncludeRelatedMatchingInfo | SzEntityIncludeRelatedRecordSummary |  
SzEntityIncludeRelatedRecordData | SzEntityIncludeInternalFeatures |  
SzEntityIncludeFeatureStats | SzIncludeFeatureScores | SzSearchIncludeStats |  
SzEntityIncludeRecordTypes | SzEntityIncludeRelatedRecordTypes |  
SzEntityIncludeRecordUnmappedData | SzIncludeMatchKeyDetails |  
SzEntityIncludeRecordFeatureDetails | SzEntityIncludeRecordFeatureStats |  
SzSearchIncludeRequest | SzSearchIncludeRequestDetails | SzEntityIncludeRecordDates
```

Field Value

[SzFlag](#)

See Also

[SzWhySearchFlags](#)

SzWhySearchDefaultFlags

The aggregate [SzFlag](#) representing the default flags for "why search" operations.

```
public const SzFlag SzWhySearchDefaultFlags = SzIncludeFeatureScores |  
SzSearchIncludeStats | SzSearchIncludeRequestDetails
```

Field Value

[SzFlag](#)

Remarks

The included [SzFlag](#) values are:

- [SzIncludeFeatureScores](#)
- [SzSearchIncludeRequestDetails](#)
- [SzSearchIncludeStats](#)

All the flags in this constant belong to the following usage groups:

- [SzWhySearchFlags](#)

See Also

<https://docs.senzing.com/flags/index.html>

Methods

FlagsToLong(SzFlag?)

Converts a nullable [SzFlag](#) to a non-null [long](#) value representation of the flag with all the appropriate bits set.

```
public static long FlagsToLong(SzFlag? flags)
```

Parameters

[flags](#) [SzFlag?](#)

Returns

[long](#)

FlagsToString(SzFlag)

Returns the [string](#) representation describing the respective [SzFlag](#) value.

```
public static string FlagsToString(this SzFlag flag)
```

Parameters

flag [SzFlag](#)

The [SzFlag](#) on which to operate.

Returns

[string](#) ↗

The [string](#) representation describing the respective [SzFlag](#) which may be an aggregate value.

FlagsToString(SzFlagUsageGroup, SzFlag?)

Formats a [string](#) representation of the specified [SzFlag](#) according to the [SzFlag](#) instances associated with the respective [SzFlagUsageGroup](#).

```
public static string FlagsToString(this SzFlagUsageGroup group, SzFlag? flag)
```

Parameters

group [SzFlagUsageGroup](#)

The [SzFlagUsageGroup](#) on which to operate.

flag [SzFlag?](#)

The [SzFlag](#) value (which may be an aggregate value) to format as a [string](#).

Returns

[string](#) ↗

The [string](#) describing the specified flags.

Remarks

NOTE: This method is useful in logging which flags were passed to a particular method using the [SzFlagUsageGroup](#) for the flags that are accepted by that method.

Some [SzFlag](#) values have the same underlying bitwise flag value which can lead to ambiguity with the default [ToString\(\)](#) implementation for enums. However, none of the [SzFlag](#) instances for a single [SzFlagUsageGroup](#) should overlap in bitwise values and this

method will prefer the [SzFlag](#) symbol belonging to the respective [SzFlagUsageGroup](#) for formatting the [string](#).

If the respective [SzFlagUsageGroup](#) is an aggregate group value representing multiple groups (or no groups) then this method simply defaults to formatting the set bit values using numeric representation, foregoing the use of symbolic flag names altogether.

FlagsToString(SzFlag?)

Returns the [string](#) representation describing the specified [SzFlag](#) value.

```
public static string FlagsToString(SzFlag? flag)
```

Parameters

[flag](#) [SzFlag?](#)

The [SzFlag](#) to format as a [string](#).

Returns

[string](#) ↗

The [string](#) representation describing the specified [SzFlag](#) which may be an aggregate value.

GetFlags(SzFlagUsageGroup)

Obtains the flags belonging to an **individual** [SzFlagUsageGroup](#) as an aggregate [SzFlag](#) value for all flags associated with the specified group.

```
public static SzFlag GetFlags(SzFlagUsageGroup group)
```

Parameters

[group](#) [SzFlagUsageGroup](#)

The **individual** [SzFlagUsageGroup](#) for which to obtain the list of flags with their names.

Returns

[SzFlag](#)

The aggregate [SzFlag](#) value with the bits set for all flags belonging to specified **individual** [SzFlagUsageGroup](#).

Exceptions

[ArgumentException](#)

If the specified [SzFlagUsageGroup](#) is itself an aggregate value of multiple groups or equal to [SzNoFlagUsageGroups](#) (i.e.: zero).

GetFlagsByName()

Obtains all individually declared flag constants from [SzFlag](#) as a read-only dictionary of individual [SzFlag](#) symbolic name keys mapped to [SzFlag](#) values.

```
public static ReadOnlyDictionary<string, SzFlag> GetFlagsByName()
```

Returns

[ReadOnlyDictionary](#)<[string](#), [SzFlag](#)>

The read-only dictionary of individual [SzFlag](#) symbolic name keys mapped to [SzFlag](#) values for all individual flag constants from the [SzFlag](#) enum.

Remarks

NOTE: This is provided for disambiguation of different flags that otherwise have the same value (e.g.: [SzSearchIncludePossiblySame](#) and [SzExportIncludePossiblySame](#)).

GetFlagsByName(SzFlagUsageGroup)

Obtains the flags belonging to an **individual** [SzFlagUsageGroup](#) as a read-only dictionary of individual [SzFlag](#) symbolic name keys mapped to [SzFlag](#) values.

```
public static ReadOnlyDictionary<string, SzFlag> GetFlagsByName(SzFlagUsageGroup
```

group)

Parameters

group [SzFlagUsageGroup](#)

The **individual** [SzFlagUsageGroup](#) for which to obtain the list of flags with their names.

Returns

[ReadOnlyDictionary](#)<[string](#), [SzFlag](#)>

A read-only dictionary of individual [SzFlag](#) symbolic name keys mapped to [SzFlag](#) values for those flags belonging to specified **individual** [SzFlagUsageGroup](#).

Remarks

NOTE: This is provided as an alternative to [GetFlags\(SzFlagUsageGroup\)](#) for disambiguation of different flags that otherwise have the same value (e.g.: [SzSearchIncludePossiblySame](#) and [SzExportIncludePossiblySame](#)).

Exceptions

[ArgumentException](#)

If the specified [SzFlagUsageGroup](#) is itself an aggregate value of multiple groups or equal to [SzNoFlagUsageGroups](#) (i.e.: zero).

GetGroups(string)

Gets the aggregate [SzFlagUsageGroup](#) value representing the one or more usage groups associated with the specified symbolic [SzFlag](#) name.

```
public static SzFlagUsageGroup GetGroups(string symbolicFlagName)
```

Parameters

symbolicFlagName [string](#)

The symbolic flag name from the [SzFlag](#) enum.

Returns

[SzFlagUsageGroup](#)

The [SzFlagUsageGroup](#) value that aggregates all the groups associated with the specified [SzFlag](#) symbolic name.

Remarks

If the specified name is not recognized as a symbolic name from the [SzFlag](#) enum then an [ArgumentException](#) is thrown.

Exceptions

[ArgumentException](#)

If the specified symbolic flag name is not recognized as an enum symbol from the [SzFlag](#) enum.

GetNamesByFlag(SzFlagUsageGroup)

Obtains the flags belonging to an **individual** [SzFlagUsageGroup](#) as a read-only dictionary of individual [SzFlag](#) keys mapped to [SzFlag](#) symbolic name values.

```
public static IReadOnlyDictionary<SzFlag, string> GetNamesByFlag(SzFlagUsageGroup group)
```

Parameters

group [SzFlagUsageGroup](#)

The **individual** [SzFlagUsageGroup](#) for which to obtain the list of flags with their names.

Returns

[ReadOnlyDictionary](#)<[SzFlag](#), [string](#)>

A read-only dictionary of individual [SzFlag](#) keys mapped to [SzFlag](#) symbolic name values for those flags belonging to specified **individual** [SzFlagUsageGroup](#).

Remarks

NOTE: This is provided as an alternative to [GetFlags\(SzFlagUsageGroup\)](#) for disambiguation of different flags that otherwise have the same value (e.g.: [SzSearchIncludePossiblySame](#) and [SzExportIncludePossiblySame](#)) so the actual flag name can be obtained with respect to a specific [SzFlagUsageGroup](#) since there is no ambiguity within a group.

Exceptions

[ArgumentException](#)

If the specified [SzFlagUsageGroup](#) is itself an aggregate value of multiple groups or equal to [SzNoFlagUsageGroups](#) (i.e.: zero).

Class SzLicenseException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Extends [SzUnrecoverableException](#) to define an exceptional condition triggered by an invalid, expired or exhausted Senzing license.

```
public class SzLicenseException : SzUnrecoverableException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← [SzUnrecoverableException](#) ← SzLicenseException

Implements

[ISerializable](#)

Inherited Members

[SzException.ErrorCode](#) , [Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

SzLicenseException()

Default constructor.

```
public SzLicenseException()
```

SzLicenseException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzLicenseException(Exception cause)
```

Parameters

cause [Exception](#)

The underlying cause for the exception.

SzLicenseException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzLicenseException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzLicenseException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzLicenseException(long? errorCode, string message, Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzLicenseException(string)

Constructs with a message explaining the reason for the exception.

```
public SzLicenseException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzLicenseException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzLicenseException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Class SzNotFoundException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Extends [SzBadInputException](#) to define an exceptional condition where the provided bad input to a Senzing operation is an identifier that could not be used to successfully locate required data for that operation.

```
public class SzNotFoundException : SzBadInputException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← [SzBadInputException](#) ← [SzNotFoundException](#)

Implements

[ISerializable](#)

Inherited Members

[SzException.ErrorCode](#) , [Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

SzNotFoundException()

Default constructor.

```
public SzNotFoundException()
```

SzNotFoundException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzNotFoundException(Exception cause)
```

Parameters

cause [Exception](#)

The underlying cause for the exception.

SzNotFoundException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzNotFoundException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzNotFoundException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzNotFoundException(long? errorCode, string message, Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzNotFoundException(string)

Constructs with a message explaining the reason for the exception.

```
public SzNotFoundException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzNotFoundException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzNotFoundException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Class SzNotInitializedException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Extends [SzUnrecoverableException](#) to define an exceptional condition triggered by Senzing not being initialized.

```
public class SzNotInitializedException : SzUnrecoverableException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← [SzUnrecoverableException](#) ← SzNotInitializedException

Implements

[ISerializable](#)

Inherited Members

[SzException.ErrorCode](#) , [Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

SzNotInitializedException()

Default constructor.

```
public SzNotInitializedException()
```

SzNotInitializedException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzNotInitializedException(Exception cause)
```

Parameters

cause [Exception](#)

The underlying cause for the exception.

SzNotInitializedException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzNotInitializedException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzNotInitializedException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzNotInitializedException(long? errorCode, string message, Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzNotInitializedException(string)

Constructs with a message explaining the reason for the exception.

```
public SzNotInitializedException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzNotInitializedException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzNotInitializedException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Interface SzProduct

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Defines the C# interface to the Senzing product functions.

```
public interface SzProduct
```

Examples

An `SzProduct` instance is typically obtained from an [SzEnvironment](#) instance via the [GetProduct\(\)](#) method as follows.

For example:

```
// How to obtain an SzProduct instance
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the product from the environment
    SzProduct product = env.GetProduct();

    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception (varies by application)
    LogError("Failed to get SzProduct.", e);
}
```

Remarks

The Senzing product functions provide information regarding the Senzing product installation and user license.

Methods

GetLicense()

Gets the details and entitlements of the applied product license.

```
string GetLicense()
```

Returns

[string](#)

The JSON document describing the license details.

Examples

Usage:

```
// How to obtain the Senzing product license JSON
try
{
    // obtain the SzEnvironment (varies by application)
    SzEnvironment env = GetEnvironment();

    // get the SzProduct instance
    SzProduct product = env.GetProduct();

    // obtain the license JSON
    string license = product.GetLicense();
    demoResult = license; // @replace
    // do something with the returned JSON (e.g.: parse it and extract values)
    JsonObject? jsonObj = JsonNode.Parse(license)?.AsObject();

    string? expiration = jsonObj?["expireDate"]?.GetValue<string>();
    int? recordLimit = jsonObj?["recordLimit"]?.GetValue<int>();

    . . .

}

catch (SzException e)
{
    // handle or rethrow the exception
    LogError("Failed to get license information.", e);
}
```

Example Result: (formatted for readability)

```
{  
    "customer": "",  
    "contract": "",  
    "issueDate": "2025-10-06",  
    "licenseType": "EVAL (Solely for non-productive use)",  
    "licenseLevel": "",  
    "billing": "",  
    "expireDate": "2026-10-07",  
    "recordLimit": 500,  
    "advSearch": 0  
}
```

Remarks

NOTE: The details do not include the license key.

Exceptions

[SzException](#)

Thrown if a failure occurs.

GetVersion()

Gets the product version details.

```
string GetVersion()
```

Returns

[string](#)

The JSON document of version details.

Examples

Usage:

```
// How to obtain the Senzing product version JSON  
try  
{
```

```

// obtain the SzEnvironment (varies by application)
SzEnvironment env = GetEnvironment();

// get the SzProduct instance
SzProduct product = env.GetProduct();

// obtain the version JSON
string result = product.GetVersion();
demoResult = result; // @replace
// do something with the returned JSON (e.g.: parse it and extract values)
JsonObject? jsonObj = JsonNode.Parse(result)?.AsObject();

string? version = jsonObj?["VERSION"]?.GetValue<string>();
string? buildDate = jsonObj?["BUILD_DATE"]?.GetValue<string>();

. . .

}

catch (SzException e)
{
    LogError("Failed to get version information.", e);
}

```

Example Result:

(formatted for readability)

```
{
  "PRODUCT_NAME": "Senzing SDK",
  "VERSION": "4.1.0",
  "BUILD_VERSION": "4.1.0.25279",
  "BUILD_DATE": "2025-10-06",
  "BUILD_NUMBER": "2025_10_06__20_46",
  "COMPATIBILITY_VERSION": {
    "CONFIG_VERSION": "11"
  },
  "SCHEMA_VERSION": {
    "ENGINE_SCHEMA_VERSION": "4.1",
    "MINIMUM_REQUIRED_SCHEMA_VERSION": "4.0",
    "MAXIMUM_REQUIRED_SCHEMA_VERSION": "4.99"
  }
}
```

Exceptions

[SzException](#)

Thrown if a failure occurs.

Class SzReplaceConflictException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Describes an exceptional condition when an attempt is made to replace a Senzing value with a new value providing it has not already been changed, however, the current value is no longer the expected value and has therefore already been changed.

```
public class SzReplaceConflictException : SzException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← SzReplaceConflictException

Implements

[ISerializable](#)

Inherited Members

[SzException.ErrorCode](#) , [Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

SzReplaceConflictException()

Default constructor.

```
public SzReplaceConflictException()
```

SzReplaceConflictException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzReplaceConflictException(Exception cause)
```

Parameters

cause [Exception](#)

The underlying cause for the exception.

SzReplaceConflictException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzReplaceConflictException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzReplaceConflictException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzReplaceConflictException(long? errorCode, string message, Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzReplaceConflictException(string)

Constructs with a message explaining the reason for the exception.

```
public SzReplaceConflictException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzReplaceConflictException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzReplaceConflictException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Class SzRetryTimeoutExceededException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Extends [SzRetryableException](#) to define an exceptional condition where an operation failed because a timeout was exceeded. Retrying the operation (possibly with a longer timeout) may result in it completing successfully.

```
public class SzRetryTimeoutExceededException : SzRetryableException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← [SzRetryableException](#) ← SzRetryTimeoutExceededException

Implements

[ISerializable](#)

Inherited Members

[SzException.ErrorCode](#) , [Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

SzRetryTimeoutExceededException()

Default constructor.

```
public SzRetryTimeoutExceededException()
```

SzRetryTimeoutExceededException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzRetryTimeoutExceededException(Exception cause)
```

Parameters

cause [Exception](#)

The underlying cause for the exception.

SzRetryTimeoutExceededException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzRetryTimeoutExceededException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzRetryTimeoutExceededException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzRetryTimeoutExceededException(long? errorCode, string message, Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzRetryTimeoutExceededException(string)

Constructs with a message explaining the reason for the exception.

```
public SzRetryTimeoutExceededException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzRetryTimeoutExceededException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzRetryTimeoutExceededException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Class SzRetryableException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Defines an exceptional condition where the failure is an intermittent condition and the operation may be retried with the same parameters with an expectation of success.

```
public class SzRetryableException : SzException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← SzRetryableException

Implements

[ISerializable](#)

Derived

[SzDatabaseConnectionLostException](#), [SzDatabaseTransientException](#),
[SzRetryTimeoutExceededException](#)

Inherited Members

[SzException.ErrorCode](#) , [Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

SzRetryableException()

Default constructor.

```
public SzRetryableException()
```

SzRetryableException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzRetryableException(Exception cause)
```

Parameters

cause [Exception](#)?

The underlying cause for the exception.

SzRetryableException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzRetryableException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzRetryableException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzRetryableException(long? errorCode, string message, Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzRetryableException(string)

Constructs with a message explaining the reason for the exception.

```
public SzRetryableException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzRetryableException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzRetryableException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Class SzUnhandledException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Extends [SzUnrecoverableException](#) to define an exceptional condition caused by an otherwise unhandled and unexpected failure in the Senzing SDK.

```
public class SzUnhandledException : SzUnrecoverableException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← [SzUnrecoverableException](#) ← SzUnhandledException

Implements

[ISerializable](#)

Inherited Members

[SzException.ErrorCode](#) , [Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

SzUnhandledException()

Default constructor.

```
public SzUnhandledException()
```

SzUnhandledException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzUnhandledException(Exception cause)
```

Parameters

cause [Exception](#)

The underlying cause for the exception.

SzUnhandledException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzUnhandledException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzUnhandledException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzUnhandledException(long? errorCode, string message, Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzUnhandledException(string)

Constructs with a message explaining the reason for the exception.

```
public SzUnhandledException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzUnhandledException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzUnhandledException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Class SzUnknownDataSourceException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Extends [SzBadInputException](#) to define an exceptional condition where the provided bad input to a Senzing operation is an identifier that could not be used to successfully locate required data for that operation.

```
public class SzUnknownDataSourceException : SzBadInputException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← [SzBadInputException](#) ← SzUnknownDataSourceException

Implements

[ISerializable](#)

Inherited Members

[SzException.ErrorCode](#) , [Exception.GetBaseException\(\)](#) ,
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#) , [Exception.GetType\(\)](#) ,
[Exception.ToString\(\)](#) , [Exception.Data](#) , [Exception.HelpLink](#) , [Exception.HResult](#) ,
[Exception.InnerException](#) , [Exception.Message](#) , [Exception.Source](#) ,
[Exception.StackTrace](#) , [Exception.TargetSite](#) , [Exception.SerializeObjectState](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

SzUnknownDataSourceException()

Default constructor.

```
public SzUnknownDataSourceException()
```

SzUnknownDataSourceException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzUnknownDataSourceException(Exception cause)
```

Parameters

cause [Exception](#)

The underlying cause for the exception.

SzUnknownDataSourceException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzUnknownDataSourceException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzUnknownDataSourceException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzUnknownDataSourceException(long? errorCode, string message,  
Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzUnknownDataSourceException(string)

Constructs with a message explaining the reason for the exception.

```
public SzUnknownDataSourceException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzUnknownDataSourceException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzUnknownDataSourceException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Class SzUnrecoverableException

Namespace: [Senzing.Sdk](#)

Assembly: Senzing.Sdk.dll

Defines an exceptional condition where the failure is not recoverable and all operations should be stopped until the system can be modified to resolve the condition causing the failure.

```
public class SzUnrecoverableException : SzException, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← [SzException](#) ← SzUnrecoverableException

Implements

[ISerializable](#)

Derived

[SzDatabaseException](#), [SzLicenseException](#), [SzNotInitializedException](#),
[SzUnhandledException](#)

Inherited Members

[SzException.ErrorCode](#), [Exception.GetBaseException\(\)](#),
[Exception.GetObjectData\(SerializationInfo, StreamingContext\)](#), [Exception.GetType\(\)](#),
[Exception.ToString\(\)](#), [Exception.Data](#), [Exception.HelpLink](#), [Exception.HResult](#),
[Exception.InnerException](#), [Exception.Message](#), [Exception.Source](#),
[Exception.StackTrace](#), [Exception.TargetSite](#), [Exception.SerializeObjectState](#),
[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#)

Constructors

SzUnrecoverableException()

Default constructor.

```
public SzUnrecoverableException()
```

SzUnrecoverableException(Exception)

Constructs with the {@link Throwable} that is the underlying cause for the exception.

```
public SzUnrecoverableException(Exception cause)
```

Parameters

cause [Exception](#)?

The underlying cause for the exception.

SzUnrecoverableException(long?, string)

Constructs with a message explaining the reason for the exception.

```
public SzUnrecoverableException(long? errorCode, string message)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

SzUnrecoverableException(long?, string, Exception)

Constructs with the Senzing error code, the message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzUnrecoverableException(long? errorCode, string message, Exception cause)
```

Parameters

errorCode [long](#)?

The underlying Senzing error code.

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

SzUnrecoverableException(string)

Constructs with a message explaining the reason for the exception.

```
public SzUnrecoverableException(string message)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

SzUnrecoverableException(string, Exception)

Constructs with a message explaining the reason for the exception and the [Exception](#) that is the underlying cause for the exception.

```
public SzUnrecoverableException(string message, Exception cause)
```

Parameters

message [string](#)

The message explaining the reason for the exception.

cause [Exception](#)

The underlying cause for the exception.

Namespace Senzing.Sdk.Core

Classes

[SzCoreEnvironment](#)

Provides the core implementation of [SzEnvironment](#) that directly initializes the Senzing SDK modules and provides management of the Senzing environment in this process.

[SzCoreEnvironment.AbstractBuilder<E, B>](#)

Provides a base class for builder implementations of [SzCoreEnvironment](#) and its derived classes.

[SzCoreEnvironment.Builder](#)

The builder class for creating an instance of [SzCoreEnvironment](#).

[SzCoreUtilities](#)

Provides utility functions that are useful when implementing the Senzing SDK interfaces from the [Senzing.Sdk](#) namespace. This is functionality leveraged by the "core" implementation that is made visible to aid in other implementations.

Interfaces

[SzCoreEnvironment.Initializer](#)

Provides an interface for initializing an instance of [SzCoreEnvironment](#).

Class SzCoreEnvironment

Namespace: [Senzing.Sdk.Core](#)

Assembly: Senzing.Sdk.dll

Provides the core implementation of [SzEnvironment](#) that directly initializes the Senzing SDK modules and provides management of the Senzing environment in this process.

```
public class SzCoreEnvironment : SzEnvironment
```

Inheritance

[object](#) ← SzCoreEnvironment

Implements

[SzEnvironment](#)

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Examples

Usage:

```
// get the settings (varies by application)
string settings = GetSettings();

// get the instance name (varies by application)
string instanceName = GetInstanceName();

// construct the environment
SzEnvironment env = SzCoreEnvironment.NewBuilder()
    .InstanceId(instanceId)
    .Settings(settings)
    .VerboseLogging(false)
    .Build();

// use the environment for some time (usually as long as the application is running)
.
```

```
// destroy the environment when done (sometimes in a finally block)
env.Destroy();
```

Constructors

SzCoreEnvironment(Initializer)

Protected constructor used by the [SzCoreEnvironment.Builder](#) to construct the instance.

```
protected SzCoreEnvironment(SzCoreEnvironment.Initializer initializer)
```

Parameters

initializer [SzCoreEnvironment.Initializer](#)

The [SzCoreEnvironment.Initializer](#) with which to construct. (typically an instance of [SzCoreEnvironment.AbstractBuilder<E, B>](#))

Fields

DefaultInstanceName

The default instance name to use for the Senzing initialization. The value is "Senzing Instance".

```
public const string DefaultInstanceName = "Senzing Instance"
```

Field Value

[string](#)

Remarks

An explicit value can be provided via the [SzCoreEnvironment.Builder](#)'s [InstanceName](#) method.

See Also

[NewBuilder\(\)](#), [SzCoreEnvironment.Builder](#), [InstanceName\(string\)](#)

DefaultSettings

The default "bootstrap" settings with which to initialize the [SzCoreEnvironment](#) when an explicit settings value has not been provided via the [SzCoreEnvironment.Builder](#)'s [Settings](#) method.

```
public const string DefaultSettings = "{ }"
```

Field Value

[string](#)

Remarks

If this is used it will initialize Senzing for access to only the [SzProduct](#) and [SzConfig](#) interfaces when Senzing is installed in the default location for the platform. The value of this constant is "{ }".

NOTE: Using these settings is only useful for accessing the [SzProduct](#) and [SzConfig](#) interfaces since [SzEngine](#), [SzConfigManager](#) and [SzDiagnostic](#) require database configuration to connect to the Senzing repository.

See Also

[NewBuilder\(\)](#), [SzCoreEnvironment.Builder](#), [Settings\(string\)](#)

Methods

Destroy()

Implemented to destroy each of the underlying core API objects.

```
public void Destroy()
```

Remarks

This method has no effect if it has previously been called for this instance.

DoExecute<T>(Func<T>)

Called by [Execute<T>\(Func<T>\)](#) after ensuring that this instance is not [destroyed](#) and ensuring any calls to [Destroy\(\)](#) will block until after this method's invocation is completed.

```
protected virtual T DoExecute<T>(Func<T> task)
```

Parameters

task [Func](#)<T>

The no-argument [Func](#) to execute.

Returns

T

The result from the specified task.

Type Parameters

T

The return type of the specified function and the type returned by this function. If your function does not really need a return value, then have it return [null](#).

Remarks

If successful, this will return the result of the specified task. If not successful, this will throw any exception produced by the specified task, wrapping it in an [SzException](#) if it is a that is not of type [SzException](#).

Execute<T>(Func<T>)

Executes the specified task ([Func](#)) via [DoExecute<T>\(Func<T>\)](#) after ensuring that this instance is not [destroyed](#) and will not be [destroyed](#) before the task's completion.

```
protected virtual T Execute<T>(Func<T> task)
```

Parameters

task [Func](#)<T>

The no-argument `Func` to execute.

Returns

`T`

The result from the specified task.

Type Parameters

`T`

The return type of the specified function and the type returned by this function. If your function does not really need a return value, then have it return `null`.

Remarks

This is used by the core implementations of [SzEngine](#), [SzProduct](#), [SzConfigManager](#), [SzConfig](#) and [SzDiagnostic](#) to execute their functionality and ensure the environment is stable during execution.

If successful, this will return the result of the specified task. If not successful, this will throw any exception produced by the specified task, wrapping it in an [SzException](#) if it is a that is not of type [SzException](#).

Exceptions

[SzException](#)

If the specified task triggers a failure.

[SzEnvironmentDestroyedException](#)

If this `SzCoreEnvironment` instance has already been destroyed.

GetActiveConfigID()

Implemented to call the underlying native method to obtain the active config ID.

```
public long GetActiveConfigID()
```

Returns

[long](#)

GetActiveInstance()

Gets the current active instance of [SzCoreEnvironment](#).

```
public static SzCoreEnvironment GetActiveInstance()
```

Returns

[SzCoreEnvironment](#)

The current active instance of [SzCoreEnvironment](#), or [null](#) if there is no active instance.

Remarks

An active instance is one that has been constructed and has not yet been destroyed. There can be at most one active instance. If no active instance exists then [null](#) is returned.

GetConfigManager()

Implemented to return the Senzing.Sdk.Core.SzCoreConfigManager associated with this instance.

```
public SzConfigManager GetConfigManager()
```

Returns

[SzConfigManager](#)

The Senzing.Sdk.Core.SzCoreConfigManager associated with this instance.

GetDiagnostic()

Implemented to return the Senzing.Sdk.Core.SzCoreDiagnostic associated with this instance.

```
public SzDiagnostic GetDiagnostic()
```

Returns

[SzDiagnostic](#)

The Senzing.Sdk.Core.SzCoreDiagnostic associated with this instance.

GetEngine()

Implemented to return the Senzing.Sdk.Core.SzCoreEngine associated with this instance.

```
public SzEngine GetEngine()
```

Returns

[SzEngine](#)

The Senzing.Sdk.Core.SzCoreEngine associated with this instance.

GetProduct()

Implemented to return the Senzing.Sdk.Core.SzCoreProduct associated with this instance.

```
public SzProduct GetProduct()
```

Returns

[SzProduct](#)

The Senzing.Sdk.Core.SzCoreProduct associated with this instance.

IsDestroyed()

Implemented to return `true` if this instance is the currently active instance, otherwise `false`.

```
public bool IsDestroyed()
```

Returns

[bool](#)

`true` if this instance is the currently active instance, otherwise `false`.

NewBuilder()

Creates a new instance of [SzCoreEnvironment.Builder](#) for setting up an instance of `SzCoreEnvironment`.

```
public static SzCoreEnvironment.Builder NewBuilder()
```

Returns

[SzCoreEnvironment.Builder](#)

The [SzCoreEnvironment.Builder](#) for configuring and initializing the `SzCoreEnvironment`.

Remarks

Keep in mind that while multiple [SzCoreEnvironment.Builder](#) instances can exists, **only one active instance** of `SzCoreEnvironment` can exist at time. An active instance is one that has not yet been destroyed.

Alternatively, you can directly call the [Builder\(\)](#) constructor.

Reinitialize(long)

Implemented to call the underlying native method to reinitialize.

```
public void Reinitialize(long configID)
```

Parameters

`configID` [long](#)

ValidateActiveInstance()

Internal method that is called by [GetActiveInstance\(\)](#) to validate the active instance before returning it.

```
protected bool ValidateActiveInstance()
```

Returns

[bool](#)

`true` if this instance is still active, or `false` if this instance has been destroyed.

Remarks

It is expected that if this instance is in the process of destroying itself, then this method will **block** until the completion of the [Destroy\(\)](#) method. `true` is returned if this instance is active and has **not** had its [Destroy\(\)](#) method called, otherwise `false` is returned for an instance that was already destroyed.

See Also

[SzEnvironment](#)

Class SzCoreEnvironment.AbstractBuilder<E, B>

Namespace: [Senzing.Sdk.Core](#)

Assembly: Senzing.Sdk.dll

Provides a base class for builder implementations of [SzCoreEnvironment](#) and its derived classes.

```
public abstract class SzCoreEnvironment.AbstractBuilder<E, B> :  
    SzCoreEnvironment.Initializer where E : SzCoreEnvironment where B :  
    SzCoreEnvironment.AbstractBuilder<E, B>
```

Type Parameters

E

The [SzCoreEnvironment](#)-derived class build by instances of this class.

B

The [SzCoreEnvironment.AbstractBuilder<E, B>](#)-derived class of the implementation.

Inheritance

[object](#) ↗ ← SzCoreEnvironment.AbstractBuilder<E, B>

Implements

[SzCoreEnvironment.Initializer](#)

Derived

[SzCoreEnvironment.Builder](#)

Inherited Members

[object.Equals\(object\)](#) ↗ , [object.Equals\(object, object\)](#) ↗ , [object.GetHashCode\(\)](#) ↗ ,
[object.GetType\(\)](#) ↗ , [object.MemberwiseClone\(\)](#) ↗ , [object.ReferenceEquals\(object, object\)](#) ↗ ,
[object.ToString\(\)](#) ↗

Remarks

This interface is not needed to use [SzCoreEnvironment](#). It is only needed if you are extending [SzCoreEnvironment](#).

This allows the derived builder to return references to its own environment class and to its own builder class rather than base classes. When extending [SzCoreEnvironment](#) you should provide an implementation of this class that is specific to your derived class.

Constructors

AbstractBuilder()

Protected default constructor.

```
protected AbstractBuilder()
```

Methods

Build()

Creates a new [SzCoreEnvironment](#) instance of type `E` based on this builder instance.

```
public abstract E Build()
```

Returns

`E`

The newly created [SzCoreEnvironment](#) instance of type `E`.

Remarks

This method will throw an [InvalidOperationException](#) if another active [SzCoreEnvironment](#) instance exists since only one active instance can exist within a process at any given time. An active instance is one that has been constructed, but has not yet been destroyed.

Exceptions

[InvalidOperationException](#)

If another active [SzCoreEnvironment](#) instance exists when this method is invoked.

ConfigID(long?)

Sets the explicit configuration ID to use to initialize the [SzCoreEnvironment](#).

```
public B ConfigID(long? configID)
```

Parameters

[configID](#) [long](#)?

The explicit configuration ID to use to initialize the [SzCoreEnvironment](#), or [null](#) if the default configuration ID from the Senzing repository should be used.

Returns

B

A reference to this instance.

Remarks

If not specified then the default configuration ID obtained from the Senzing repository is used.

GetConfigID()

Gets the explicit configuration ID (if any) with which to initialize the [SzCoreEnvironment](#).

```
public long? GetConfigID()
```

Returns

[long](#)?

The explicit configuration ID with which to initialize the [SzCoreEnvironment](#), or [null](#) if no explicit configuration ID has been provided and the the default configuration ID from the Senzing repository should be used.

Remarks

This returns `null` if no explicit configuration ID has been provided and the default configuration ID from the Senzing repository should be used.

GetInstanceName()

Gets the Senzing instance name with which to initialize the [SzCoreEnvironment](#).

```
public string GetInstanceName()
```

Returns

[string](#)

The Senzing instance name with which to initialize the [SzCoreEnvironment](#).

GetSettings()

Gets the Senzing settings which which to initialize the [SzCoreEnvironment](#).

```
public string GetSettings()
```

Returns

[string](#)

The Senzing settings which which to initialize the [SzCoreEnvironment](#).

InstanceName(string)

Provides the Senzing instance name to initialize the [SzCoreEnvironment](#)

```
public B InstanceName(string instanceName)
```

Parameters

instanceName [string](#)

The instance name to initialize the [SzCoreEnvironment](#) or `null` or empty-string to restore the default.

Returns

B

A reference to this instance

Remarks

Call this method to override the default value of [DefaultInstanceName](#)

See Also

[DefaultInstanceName](#)

IsVerboseLogging()

Checks if initializing the [SzCoreEnvironment](#) with verbose logging.

```
public bool IsVerboseLogging()
```

Returns

[bool](#)

`true` if verbose logging will be enabled, otherwise `false`.

Settings(string)

Provides the Senzing settings to initialize the [SzCoreEnvironment](#).

```
public B Settings(string settings)
```

Parameters

`settings` [string](#)

The Senzing settings, or `null` or empty-string to restore the default value.

Returns

B

A reference to this instance.

Remarks

If this is set to `null` or empty-string then [DefaultSettings](#) will be used to provide limited functionality.

See Also

[DefaultSettings](#)

VerboseLogging(bool)

Sets the verbose logging flag for initializing the [SzCoreEnvironment](#).

```
public B VerboseLogging(bool verboseLogging)
```

Parameters

`verboseLogging` [bool](#)

`true` if verbose logging should be enabled, otherwise `false`.

Returns

B

A reference to this instance

Remarks

Call this method to explicitly set the value. If not called, the default value is `false`.

Class SzCoreEnvironment.Builder

Namespace: [Senzing.Sdk.Core](#)

Assembly: Senzing.Sdk.dll

The builder class for creating an instance of [SzCoreEnvironment](#).

```
public class SzCoreEnvironment.Builder :  
    SzCoreEnvironment.AbstractBuilder<SzCoreEnvironment, SzCoreEnvironment.Builder>,  
    SzCoreEnvironment.Initializer
```

Inheritance

[object](#) ←

[SzCoreEnvironment.AbstractBuilder](#)<[SzCoreEnvironment](#), [SzCoreEnvironment.Builder](#)> ←
[SzCoreEnvironment.Builder](#)

Implements

[SzCoreEnvironment.Initializer](#)

Inherited Members

[SzCoreEnvironment.AbstractBuilder](#)<[SzCoreEnvironment](#),
[SzCoreEnvironment.Builder](#)>.Settings(string) ,
[SzCoreEnvironment.AbstractBuilder](#)<[SzCoreEnvironment](#),
[SzCoreEnvironment.Builder](#)>.GetSettings() ,
[SzCoreEnvironment.AbstractBuilder](#)<[SzCoreEnvironment](#),
[SzCoreEnvironment.Builder](#)>.InstanceName(string) ,
[SzCoreEnvironment.AbstractBuilder](#)<[SzCoreEnvironment](#),
[SzCoreEnvironment.Builder](#)>.GetInstanceName() ,
[SzCoreEnvironment.AbstractBuilder](#)<[SzCoreEnvironment](#),
[SzCoreEnvironment.Builder](#)>.VerboseLogging(bool) ,
[SzCoreEnvironment.AbstractBuilder](#)<[SzCoreEnvironment](#),
[SzCoreEnvironment.Builder](#)>.IsVerboseLogging() ,
[SzCoreEnvironment.AbstractBuilder](#)<[SzCoreEnvironment](#),
[SzCoreEnvironment.Builder](#)>.ConfigID(long?) ,
[SzCoreEnvironment.AbstractBuilder](#)<[SzCoreEnvironment](#),
[SzCoreEnvironment.Builder](#)>.GetConfigID() ,
[SzCoreEnvironment.AbstractBuilder](#)<[SzCoreEnvironment](#),
[SzCoreEnvironment.Builder](#)>.Build() ,
[object.Equals](#)([object](#)) , [object.Equals](#)([object](#), [object](#)) , [object.GetHashCode](#)() ,

[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

Builder()

Default constructor.

```
public Builder()
```

Methods

Build()

Creates a new [SzCoreEnvironment](#) instance based this **Builder** instance.

```
public override SzCoreEnvironment Build()
```

Returns

[SzCoreEnvironment](#)

The newly created [SzCoreEnvironment](#) instance.

Remarks

This method will throw an [InvalidOperationException](#) if another active [SzCoreEnvironment](#) instance exists since only one active instance can exist within a process at any given time. An active instance is one that has been constructed, but has not yet been destroyed.

Exceptions

[InvalidOperationException](#)

If another active [SzCoreEnvironment](#) instance exists when this method is invoked.

Interface SzCoreEnvironment.Initializer

Namespace: [Senzing.Sdk.Core](#)

Assembly: Senzing.Sdk.dll

Provides an interface for initializing an instance of [SzCoreEnvironment](#).

```
public interface SzCoreEnvironment.Initializer
```

Remarks

This interface is not needed to use [SzCoreEnvironment](#). It is only needed if you are extending [SzCoreEnvironment](#).

This is provided for derived classes of [SzCoreEnvironment](#) to initialize their super class and is typically implemented by extending [SzCoreEnvironment.AbstractBuilder<E, B>](#) in creating a derived builder implementation.

Methods

GetConfigID()

Gets the explicit configuration ID (if any) with which to initialize the [SzCoreEnvironment](#).

```
long? GetConfigID()
```

Returns

[long](#)?

The explicit configuration ID with which to initialize the [SzCoreEnvironment](#), or [null](#) if no explicit configuration ID has been provided and the default configuration ID from the Senzing repository should be used.

Remarks

This returns [null](#) if no explicit configuration ID has been provided and the default configuration ID from the Senzing repository should be used.

GetInstanceName()

Gets the Senzing instance name with which to initialize the [SzCoreEnvironment](#).

```
string GetInstanceName()
```

Returns

[string](#)

The Senzing instance name with which to initialize the [SzCoreEnvironment](#).

GetSettings()

Gets the Senzing settings which which to initialize the [SzCoreEnvironment](#).

```
string GetSettings()
```

Returns

[string](#)

The Senzing settings which which to initialize the [SzCoreEnvironment](#).

IsVerboseLogging()

Checks if initializing the [SzCoreEnvironment](#) with verbose logging.

```
bool IsVerboseLogging()
```

Returns

[bool](#)

`true` if verbose logging will be enabled, otherwise `false`.

Class SzCoreUtilities

Namespace: [Senzing.Sdk.Core](#)

Assembly: Senzing.Sdk.dll

Provides utility functions that are useful when implementing the Senzing SDK interfaces from the [Senzing.Sdk](#) namespace. This is functionality leveraged by the "core" implementation that is made visible to aid in other implementations.

```
public static class SzCoreUtilities
```

Inheritance

[object](#) ← SzCoreUtilities

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) ,
[object.ToString\(\)](#)

Remarks

None of the functionality provided here is required to use the [SzCoreEnvironment](#) class, but may be useful if creating an alternate implementation of [SzEnvironment](#).

Methods

CreateConfigComment(string)

Produce an auto-generated configuration comment for the configuration manager registry. This is useful when implementing the [RegisterConfig\(string\)](#) function.

```
public static string CreateConfigComment(string configDefinition)
```

Parameters

`configDefinition` [string](#)

The `string` configuration definition.

Returns

[string](#)

The auto-generated comment, which may be empty-string if an auto-generated comment could not otherwise be inferred.

CreateSzException(long, string)

Creates the appropriate [SzException](#) instance for the specified error code.

```
public static SzException CreateSzException(long errorCode, string message)
```

Parameters

errorCode [long](#)

The error code to use to determine the specific type for the [SzException](#) instance.

message [string](#)

The error message to associate with the exception.

Returns

[SzException](#)

Remarks

If there is a failure in creating the [SzException](#) instance, then a generic [SzException](#) is created with the specified parameters and "caused by" exception describing the failure.