# Checkmarx CxEnterprise CxQuery API Guide

**V9.6.1**

**February 2023**

# Table of Contents

# 1 Preface

The CxQuery API Guide documents the Checkmarx Query Language (CxQL) used in CxAudit to query source code.

CxQL allows us to virtually data-mine any aspect of the source, and to build custom queries.

Checkmarx-provided queries are written using the CxQL.

These queries can be inherited, expanded, or rewritten.

**Note:** CxQL queries are language-dependent.

# 2 Introduction

A query written in Checkmarx Query Language allows us to analyze the scanned code and return a list of results.

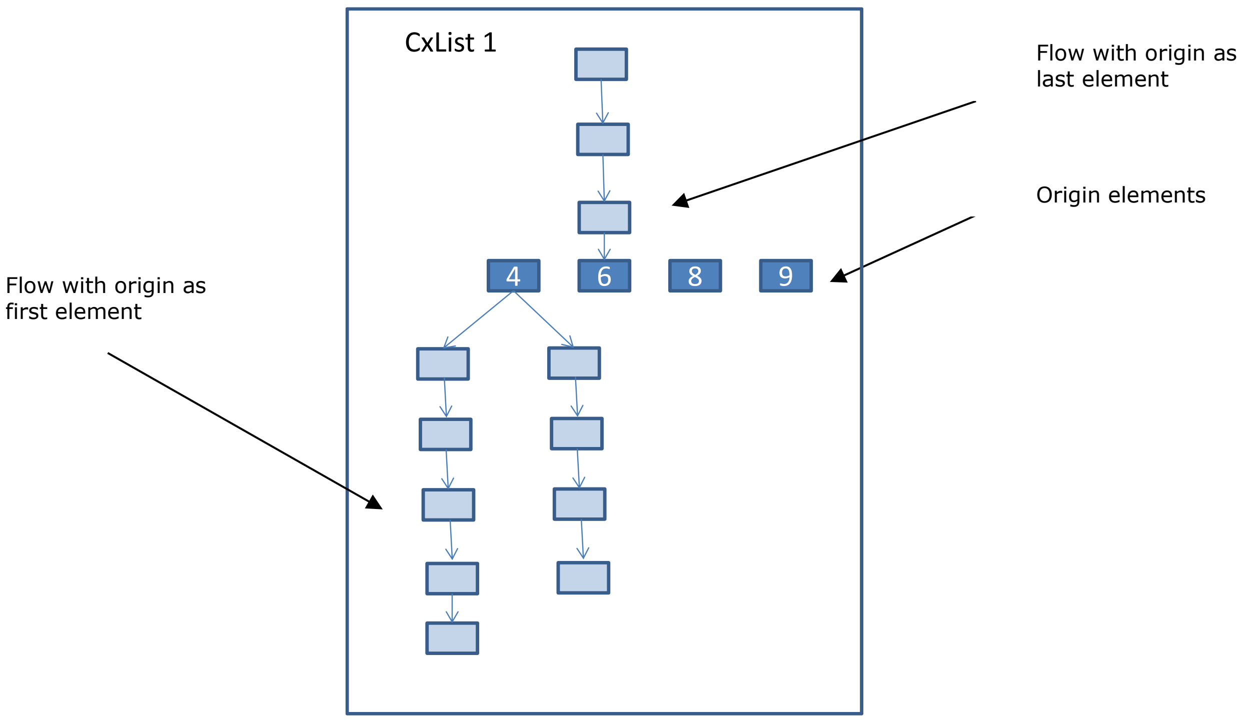
Each result can be an element in the scanned code (e.g. a variable) or a “flow” – a path in the code consisting of an ordered list of these elements.

## 2.1 Definitions

***Basic code element*** – Code elements such as variables, method invocations and assignments that have representation in the code model.

***Data flow (flow)*** – an ordered list of code elements that represent a possible data change progression in the program from a certain location where the data has changed and the end location where that change had an affect (as a subsequent data change).
Every flow is attached to an origin basic code element. This origin element may be the first or last code element in the flow. The origin element appears as the first element in the flow if that element was queried as to whether it influenced other elements. The origin element appears as the last element in the flow if that element was queried as to whether it was influenced by other elements.

***CxList*** - the central data type in CxQL. The CxList is a list that consists of basic code elements such as variables, method invocations, assignments, and so forth. Each element may have an attached flow, if the element was added to the CxList because it fulfilled a certain flow query.


CxList 1
Flow with origin as last element
Origin elements
Flow with origin as first element
4
6
8
9

There are two special CxList objects by default:

***All*** – contains all elements in the scanned code, and

***result*** – the return value of the query.

**Notes:**

- “All” contains only basic code elements (without any flow).
- To create a new empty CxList use All.NewCxList().

## 2.2 Queries and Commands:

Now we are ready for our first query:

```
CxQL

 This example demonstrates the use of "All" and "result" objects
result = All;

 This would return a list of all objects in the code for a specific language
```

CxList includes a vast assortment of commands. In the following example, we investigate the CxList FindByName command:

```
CxQL

result = All.FindByName("*MyName*")
 This would return a list of all objects where their name
 contains the string "MyName".

 It is the same as:
CxList cml = All.FindByName("*MyName*");
result = cml;
```

Because the return value of almost every command is also of type CxList, several commands can be executed consecutively as shown in the example below.

It is important to note that most CxList methods return a subset of the original CxList (we can think of the method as a **filter**).

So in the example below, consisting of chained method calls:

```
All.FindByName("*.MyName").FindByType (typeof(MemberAccess))
```

The order of execution is:

1 Return a CxList consisting of a subset of “All” (all elements in code) with name containing MyName .

2 Return a subset of the previous result , only those of type MemberAccess.

```
CxQL

result = All.FindByName("*.MyName").FindByType (typeof(MemberAccess));
 This would return a list of all access data members in the code whose name
contains the string "MyName" (e.g. a = b.MyName ).
 First we find all objects whose name ends with ".MyName", and on the result we
execute another command that retrieves only access members.
 This is the same as the following:

result = All.FindByType (typeof(MemberAccess)).FindByName("*.MyName");
* The difference is in efficiency. We want to work on the smallest groups
possible, so actually first looking by name and then by type should be more
efficient.
```

While the result in both cases is identical (order of filtering doesn’t matter), the choice of execution order can have a noticeable effect on performance.

## 2.2.1 Data Flow Graph

We have seen in the previous section several commands that can operate on CxList objects. All the commands were “static” since they locate elements in the code, but they do not capture the flow between elements. The Data Flow Graph (DFG) in Source Code Analysis (SCA) describes how data is flowed through the program. Object A is "data influenced by" object B if the value of B flows to A.

In the example below, **d** is "data influenced by" **a** and **b**, but not by **c**. This means that both **a** and **b** are "data influencing on" **d**, but not on **c**.

```
C#

a = 5;
b = 6;
c = 7;
d = a + b;
```

# 3 Using cxLog

The cxlog object is a way to output debug messages from within a query, in the CxAudit environment.

The messages can be seen in the CxAudit bottom window, in the tab named “Debug Messages”.

The most common case is when exceptions happen, so that the exception details can be viewed after the query has finished.

## For example:

```
CXQL
foreach (CxList rt in redirectThings)
{
	try
	{
		[…]
	}
	catch (Exception ex)
	{
		// in case of an exception in the loop
		cxLog.WriteDebugMessage(ex);
	}
}
```

It can also be used for more detailed inspection of the query behavior from within the query itself.

## For example:

```
CXQL
if(hexEquiv != "")
{
	cxLog.WriteDebugMessage("hexEquiv=" + hexEquiv + ", #finds=" + 
finds.Count);
	count++;
}
else
{
	cxLog.WriteDebugMessage("hexEquiv=empty" + hexEquiv + ", #finds=" + 
finds.Count);
}
```

Note that cxLog.WriteDebugMessage cannot display CxList data directly. Executing that **cxLog.WriteDebugMessage(myCxList)** will yield just an integer value.

However, in many cases (when the CxList does not represent a path), one can retrieve and output the CxList element fields.

## For example:

```
CXQL
CxList inputs = Find_Inputs();
foreach (CxList inp in inputs)
{
	CSharpGraph inp_Graph = inp.data.GetByIndex(0) as CSharpGraph;
	String inp_Name = inp_Graph.ShortName;
	cxLog.WriteDebugMessage("Name = " + inp_Name);
}
```

# 4 Using cxEnv

The cxEnv object is a way to allow access using System.IO API for environment variables within a query, in the CxAudit environment.

Since there are many differences between, for instance, Windows and Linux, this cxEnv will determine automatically your OS and, therefore, output the correct result.

There are several possibilities to find environment variables such as File line termination, Path separator, Directory separator, Volume separator and.

## For example:

```
CXQL
const string JS_EXT = ".js";
char PATH_SEP = cxEnv.DirectorySeparatorChar;
char vSeparator = cxEnv.VolumeSeparatorChar;
string controllerPath = directory + vSeparator + PATH_SEP + onlyName + PATH_SEP
+ JS_EXT;
```

In the above example, instead of declaring const string PATH_SEP = “\\”, it is possible to invoke the correct method that returns the desired output. Additionally, it’s conceivable as well to declare the volume separator as cxEnv.VolumeSeparatorChar as an alternative for const string vSeparator = “:”.

The controllerPath output will be, for example, C:\Users\User\Documents\Tests\Test.js
where C match directory variable, : is the representation for vSeparator, \ match PATH_SEP, onlyName represents already a combined path for the directory and, finaly, JS_EXT substitute .js type.
If the above example was running on a Linux machine, the output could be, for instance, /home/User/Documents/Tests/Test.js

For each possible environment variable, regarding different OS, we can have the following outputs:

- **cxEnv.NewLine** - Gets the newline string defined for the environment.
  Important remark: the property value of NewLine is a constant customized specifically for the current platform and implementation of the .NET Framework.
- **cxEnv.Path.PathSeparator** - On Windows-based desktop platforms, the value of this field is a semicolon (;) by default, but might vary on other platforms;
- **cxEnv.Path.DirectorySeparatorChar** - On Windows, the directory separator is \. On UNIX is /
- **cxEnv.Path.VolumeSeparatorChar** - The value of this field is a colon (:) on Windows and Macintosh, and a slash (/) on UNIX operating systems. This is most useful for parsing paths such as "c:\windows" or "MacVolume:System Folder";
- **cxEnv.Path.Combine()** - If necessary, use Directory separator or method Combine. This method is intended to concatenate individual strings into a single string that represents a file path.

Note that this new API was created similarly to the .NET **System** package. As such, the cxEnv.Path correlates to the **System.IO.Path** package and all its properties are devired directly. Only the cxEnv.NewLine differs from the previous ones since it devires from **System.Environment** package.

# 5 Methods Documentation

## 5.1 CxList.NewCxList Method ()

Create new empty CxList.

### Syntax

```
CxQL
public CxList NewCxList()
```

**Parameters**

N/A

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Example

```
CxQL

 This example demonstrates the CxList.NewCxlist() method.
 The input source code is:

 int b, a = 5;
 if (a == 33)
      b = 6;

CxList list_a = All.NewCxList();
list_a.Add(All.FindByName("A"));
CxList list_b = All.FindByName("b");
list_a.Add(list_b);
result = list_a;
 The resulting list will contain 4 elements
```

## 5.2 CxList.Add Method (int, IGraph)

Adds to the current instance the given graph node, indexed by the given id.

### Syntax

```
CxQL
public void Add(int id, IGraph node)
```

**Parameters**

**Id**

Id of the node to be added to the graph node.

**Node**

Graph node to be associated to the given Id.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | parameter is a null reference |

### Example

```
CxQL

 This example demonstrates the CxList.Add() method.
 The input source code is:

 int b, a = 5;
 if (a == 33)
      b = 6;

CxList myList = All.FindByName(“a”);
CSharpGraph nodeGraph = All.FindByName(“b”).GetFirstGraph();
myList.Add(nodeGraph.NodeId, nodeGraph);
result = myList;
   The resulting list will include the initial two “a”’s and the first b
```

## 5.3 CxList.Add Method (CxList)

Add all the elements from the given CxList to the current instance.

### Syntax

```
CxQL
public void Add(CxList list)
```

**Parameters**

**list**

The CxList to be added to the current CxList instance.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Example

```
CxQL

 This example demonstrates the CxList.Add() method.
 The input source code is:

 int b, a = 5;
 if (a == 33)
      b = 6;

CxList list_a = All.FindByName(“a”);
CxList list_b = All.FindByName(“b”);
list_a.Add(list_b);
result = list_a;
 The resulting list will contain 4 elements
```

## 5.4 CxList.Add Method (KeyValuePair<int, IGraph>)

Add the given pair to the current CxList instance.

### Syntax

```
CxQL
public void Add(KeyValuePair<int, IGraph> dictionary)
```

**Parameters**

**dictionary**

Pair to be added to the current CxList instance.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Example

```
CxQL

 This example demonstrates the CxList.Add() method.
 The input source code is:

 int b, a = 5;
 if (a == 33)
      b = 6;

CxList myList = All.FindByName("a");
foreach(KeyValuePair<int,IGraph> entry in All.FindByName("b"))
{
       myList.Add(entry);
}
result = myList;
 The resulting list will contain 4 elements
```

## 5.5 CxList.AddRange Method (IEnumerable<CxList>)

Adds to the current instance all the elements from the given CxLists.

### Syntax

```
CxQL
public void Add(IEnumerable<CxList> lists)
```

**Parameters**

**lists**

A list of CxLists

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | parameter is a null reference |

### Example

```
CxQL
```

```
 This example demonstrates the CxList.AddRange(IEnumerable<CxList>) method.
 The input source code is:

 int b, a = 5;
 if (a == 33)
      b = 6;

CxList a = All.FindByName(“a”);
CxList b = All.FindByName(“b”);
CxList integers = All.FindByType(typeof(IntegerLiteral));
Result.AddRange(new List<CxList>(){a, b, integers});
   The resulting list will include the two “a”’s, two “b”’s and the numbers 5,
33, 6.
```

### Version Information

Supported from 9.1.0

## 5.6 CxList.Add Method (params CxList[])

Adds to the current instance all the elements from the given CxLists.

### Syntax

```
CxQL
public void Add(params CxList[] lists)
```

**Parameters**

**lists**

An array of CxLists

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | parameter is a null reference |

### Example

```
CxQL

 This example demonstrates the CxList.Add(params CxList[]) method.
 The input source code is:

 int b, a = 5;
 if (a == 33)
      b = 6;

CxList a = All.FindByName(“a”);
CxList b = All.FindByName(“b”);
CxList integers = All.FindByType(typeof(IntegerLiteral));
Result.Add(a, b, integers);
   The resulting list will include the two “a”’s, two “b”’s and the numbers 5,
33, 6.
```

### Version Information

Supported from 9.1.0

## 5.7 CxList.CallingMethodOfAny Method (CxList)

Returns a CxList which is a subset of “this” instance and are methods or constructors declarations which matches the given CxList elements.

### Syntax

```
CxQL
public CxList CallingMethodOfAny(CxList elements)
```

**Parameters**

**elements**

The list of elements containing the methods or constructors to look for their declaration.

**Return Value**

The methods or constructor declarations which matches the given CxList elements.

### Example

```
CxQL

 This example demonstrates the CxList.CallingMethodOfAny() method.
 The input source code is:
 void foo()
 {
      int goo = 3;
      int boo = 5;
 }

result = All.CallingMethodOfAny(All.FindByName ("oo"));

 The result would consist of 1 item:
      foo (in void foo())
```

## 5.8 CxList.Clear Method ()

Clears the information in “this” instance.

### Syntax

```
CxQL
public bool Clear()
```

**Parameters**

None

**Return Value**

None

### Comments

This method removes all the information stored in the List.

### Example

```
CxQL

 This example demonstrates the CxList.Clear() method.

CxList MyList = All;
MessageBox.Show(MyList.Count.ToString());

MyList.Clear();
```

```
MessageBox.Show(MyList.Count.ToString());
```

## 5.9 CxList Concatenate Methods

### 5.9.1 CxList.Concatenate Method (CxList list, bool _testFlow)

Concatenates two nodes into a flow.

#### Syntax

```
CxQL
public CxList Concatenate (CxList list, bool _testFlow)
```

**Parameters**

**list**

A CxList containing one node only. This node will be concatenated to **this** instance

**_testFlow**

If true, searches for a flow between **this** instance and **list**. Otherwise, connects the two nodes directly (more efficient).

**Return Value**

A flow that starts with **this** instance node, and ends with the **list** parameter node.

#### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

#### Remarks

1. If either **this** instance or **list** parameter contains more than one node or contains flows, the function return value is undefined.
2. This function is deprecated, use **ConcatenatePath** instead.

#### Example

The following code example shows how you can use the Concatenate method.

```
CxQL

void main()
{
       int a = 1;
       int b = 2;
       int c = a + b;
       printf("%d", c);
}

CxList one = All.FindByName("1");
CxList two = All.FindByName("2");
result = one.Concatenate(two);

 the result would be -
      1 flow found:
            [1] -> [2]
```

#### Version Information

Supported from v7.1.2

## 5.9.2 CxList.Concatenate Method (CxList list)

Concatenates two nodes into a flow.

### Syntax

```
CxQL
public CxList Concatenate (CxList list)
```

**Parameters**

**list**

A CxList containing one node only. This node will be concatenated to **this** instance

**Return Value**

A flow that starts with **this** instance node, and ends with the **list** parameter node.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

1. This function calls CxList.Concatenate(list, false).
2. If either this instance or list parameter contains more than one node or contains flows, the function return value is undefined.
3. This function is deprecated, use ConcatenatePath instead.

### Version Information

Supported from v7.1.2

## 5.9.3 CxList.ConcatenatePath Method (CxList list, bool _testFlow)

Concatenates two flows into one connected flow.

### Syntax

```
CxQL
public CxList ConcatenatePath (CxList list, bool _testFlow)
```

**Parameters**

**list**

A CxList containing one flow only. This flow will be concatenated to **this** instance

**_testFlow**

If true, searches for a flow between **this** instance and **list**. Otherwise, connects the two flows directly (more efficient).

**Return Value**

A flow that starts with **this** instance flow, and ends with the **list** parameter flow.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

Both **this** instance and **list** have to contain only one flow (or one node as a private case), otherwise return value is undefined.

### Example

The following code example shows how you can use the ConcatenatePath method.

```
CxQL

void main()
{
	int a = 1;
	int b = 2;
}

CxList one = All.FindByName("1");
CxList a = All.FindByShortName("a").FindByType(typeof(Declarator)); //Declarator
is a new type defined in cxQL
CxList flow1 = a.InfluencedBy(one); // [1] -> [a]

CxList two = All.FindByName("2");
CxList b = All.FindByShortName("b").FindByType(typeof(Declarator));
CxList flow2 = b.InfluencedBy(two); // [2] -> [b]

result = flow2.ConcatenatePath(flow1);

 the result would be -
      1 flow found:
            [2] -> [b] -> [1] -> [a]
```

### Version Information

Supported from v7.1.2

## 5.9.4 CxList.ConcatenatePath Method (CxList list)

Concatenates two flows into one connected flow.

### Syntax

```
CxQL
public CxList ConcatenatePath (CxList list)
```

**Parameters**

**list**

A CxList containing one flow only. This flow will be concatenated to **this** instance

**Return Value**

A flow that starts with **this** instance flow, and ends with the **list** parameter flow.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

1. This function calls CxList.ConcatenatePath(list, true).
2. Both this instance and list have to contain only one flow (or one node as a private case), otherwise return value is undefined.

### Version Information

Supported from v7.1.2

## 5.9.5 CxList.ConcatenateAllPaths Method (CxList list, bool _testFlow)

Concatenates all flows in this instance to all flows in **list**.

### Syntax

```
CxQL
public CxList ConcatenateAllPaths (CxList list, bool _testFlow)
```

**Parameters**

**list**

A CxList containing flows. These flow will be concatenated to the flows in **this** instance

**_testFlow**

If true, searches for a flow between **this** instance and **list**. Otherwise, connects the two flows directly (more efficient).

**Return Value**

A product of all flows in **this** instance with the ones in **list** parameter.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

If **this** instance contains ***n*** flows in it and **list** contains ***m*** flows in it, the return set will contain ***nxm*** flows, where each flow from **this** instance will be concatenated to each flow from **list**.

### Example

The following code example shows how you can use the ConcatenateAllPaths method.

```
CxQL

void main()
{
       int a = 1;
       int b = 2;
}

CxList one = All.FindByName("1");
CxList a = All.FindByShortName("a").FindByType(typeof(Declarator));
CxList flow1 = a.InfluencedBy(one); // [1] -> [a]
CxList two = All.FindByName("2");
CxList b = All.FindByShortName("b").FindByType(typeof(Declarator));
CxList flow2 = b.InfluencedBy(two); // [2] -> [b]
CxList flow = flow1 + flow2;
result = flow.ConcatenateAllPaths(flow);
 the result would be -
      4 flow found:
            [1] -> [a] -> [1] -> [a]
              [1] -> [a] -> [2] -> [b]
              [2] -> [b] -> [1] -> [a]
              [2] -> [b] -> [2] -> [b]
```

### Version Information

Supported from v7.1.2

## 5.9.6 CxList.ConcatenateAllPaths Method (CxList list)

Concatenates all flows in this instance to all flows in **list**.

### Syntax

```
CxQL
public CxList ConcatenateAllPaths (CxList list)
```

**Parameters**

**list**

A CxList containing flows. These flow will be concatenated to the flows in **this** instance

**Return Value**

A product of all flows in **this** instance with the ones in **list** parameter.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

1. This function calls CxList.ConcatenateAllPaths(list, true).
2. If this instance contains *n* flows in it and list contains *m* flows in it, the return set will contain *nxm* flows, where each flow from this instance will be concatenated to each flow from list.

### Version Information

Supported from v7.1.2

## 5.9.7 CxList.ConcatenateAllSources Method (CxList list)

Concatenates the node in **list** to each node in **this** instance. Concatenation is node-to-node (doesn't support connecting flows).
Note: Currently is identical to calling ConcatenateAllSources with testFlow = false

### Syntax

```
CxQL
public CxList ConcatenateAllSources (CxList list)
```

**Parameters**

**list**

A CxList. It will be concatenated to each node in this instance

**Return Value**

Flows that starts with **this** instance nodes, and end with the **list** parameter node.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

1. If the **list** parameter contains more than one node or contains flows or **this** instance contains flows, the function return value is undefined.
2. The number of the returned items is same as the number of items in **this** instance.
3. This function calls the Concatenate function for each item in **this** instance with **list** as parameter.
4. Currently is identical to calling ConcatenateAllSources with testFlow = false

### Example

The following code example shows how you can use the ConcatenateAllSources method.

```
CxQL

void main()
{
       int a = 1;
       int b = 2;
}

CxList a = All.FindByShortName("a").FindByType(typeof(Declarator));
CxList b = All.FindByShortName("b").FindByType(typeof(Declarator));
CxList main = All.FindByShortName("main");
CxList list = a + b;
result = list.ConcatenateAllSources(main);

 the result would be -
      2 flow found:
            [a] -> [main]
              [b] -> [main]
```

### Version Information

Supported from v7.1.2

## 5.9.8 CxList.ConcatenateAllSources Method (CxList list, bool testFlow)

Concatenates the node in **list** to each node in **this** instance. Concatenation is node-to-node (doesn't support connecting flows).

### Syntax

```
CxQL
public CxList ConcatenateAllSources (CxList list, bool testFlow)
```

**Parameters**

**list**

A CxList. It will be concatenated to each node in this instance

**testFlow**

If this parameter true -> test possible flow , otherwise connect directly

**Return Value**

Flows that starts with **this** instance nodes, and end with the **list** parameter node.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

1. If the **list** parameter contains more than one node or contains flows or **this** instance contains flows, the function return value is undefined.
2. The number of the returned items is same as the number of items in **this** instance.
3. This function calls the Concatenate function for each item in **this** instance with **list** as parameter.

### Example

The following code example shows how you can use the ConcatenateAllSources method.

```
CxQL

void main()
{
	int a = 1;
	int b = 2;
}

CxList a = All.FindByShortName("a").FindByType(typeof(Declarator));
CxList b = All.FindByShortName("b").FindByType(typeof(Declarator));
CxList main = All.FindByShortName("main");
CxList list = a + b;
result = list.ConcatenateAllSources(main, false);

 the result would be -
      2 flow found:
            [a] -> [main]
              [b] -> [main]
```

### Version Information

Supported from v7.1.2

## 5.9.9 CxList.ConcatenateAllTargets Method (CxList list)

Concatenates each node in the **list** to the node in **this** instance. Concatenation is node-to-node (doesn't support connecting flows).

### Syntax

```
CxQL
public CxList ConcatenateAllTargets (CxList list)
```

**Parameters**

**list**

A CxList. It will be concatenated to each node in this instance

**Return Value**

Flows that start with this instance nodes, and end with the **list** parameter node

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

1. If the "**this**" instance parameter contains more than one node or contains flows or **list** contains flows, the function return value is undefined.
2. The number of the returned items is same as the number of items in **list**.
3. This function calls the Concatenate function for **this** instance with each item in **list** as parameter..
4. Currently is identical to calling ConcatenateAllTargets with testFlow = false

### Example

The following code example shows how you can use the ConcatenateAllSources method.

```
CxQL

void main()
{
	int a = 1;
	int b = 2;
}

CxList a = All.FindByShortName("a").FindByType(typeof(Declarator));
CxList b = All.FindByShortName("b").FindByType(typeof(Declarator));
CxList main = All.FindByShortName("main");
CxList list = a + b;
result = main.ConcatenateAllTargets(list);

 the result would be -
      2 flow found:
            [main] -> [a]
              [main] -> [b]
```

### Version Information

Supported from v7.1.2

## 5.9.10 CxList.ConcatenateAllTargets Method (CxList list, bool testFlow)

Concatenates each node in the **list** to the node in **this** instance. Concatenation is node-to-node (doesn't support connecting flows).

### Syntax

```
CxQL
public CxList ConcatenateAllTargets (CxList list, bool testFlow)
```

**Parameters**

**list**

A CxList. It will be concatenated to each node in this instance

**testFlow**

If this parameter true -> test possible flow , otherwise connect directly

**Return Value**

Flows that start with this instance nodes, and end with the **list** parameter node

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

1. If the "**this**" instance parameter contains more than one node or contains flows or **list** contains flows, the function return value is undefined.
2. The number of the returned items is same as the number of items in **list**.
3. This function calls the Concatenate function for **this** instance with each item in **list** as parameter..

### Example

The following code example shows how you can use the ConcatenateAllSources method.

```
CxQL

void main()
{
       int a = 1;
       int b = 2;
}

CxList a = All.FindByShortName("a").FindByType(typeof(Declarator));
CxList b = All.FindByShortName("b").FindByType(typeof(Declarator));
CxList main = All.FindByShortName("main");
CxList list = a + b;
result = main.ConcatenateAllTargets(list, false);

 the result would be -
      2 flow found:
            [main] -> [a]
              [main] -> [b]
```

### Version Information

Supported from v7.1.2

## 5.10 CxList.Contained Method (CxList, GetStartEndNodesType)

Returns a subset of "this" instance whose elements are contained in the given list, filtered according to the given nodes type.

### Syntax

```
CxQL
public CxList Contained(CxList pathList, GetStartEndNodesType requestedType)
```

**Parameters**

**pathList**

The list where the method looks for the requested node type.

**requestedType**

An enum matching the relevant GetStartEndNodes types, which are:

EndNodesOnly, StartNodesOnly, StartAndEndNodes, AllNodes and AllButNotStartAndEnd

**Return Value**

A subset of "this" instance with elements from the requested nodes type.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.Contained() method.
 The input source code is:

 void foo()
{
       int b = 2, a = 5, c;
       if (a > b)
              b = 3;
       c = b;
}
```

```
result =
All.FindByShortName("b").Contained(All.InfluencedBy(All.FindById(50)),
CxList.GetStartEndNodesType.AllNodes); //Id 50 is "3" in "b = 3;"

 The result would consist of 2 items:
           b (from b = 3;)
           b (from c = b;)

result = All.FindByShortName(“a”).Contained(All.InfluencedBy(All.FindById(50)),
GetStartEndNodesType.EndNodesOnly); //Id 50 is "3" in "b = 3;"

 The result would consist of 0 items
```

## 5.11 CxList.CxSelectDomProperty<T> Method (Func<T,IGraph) where T : CSharpGraph

Returns a new CxList that includes selected property that exists in DOM type T and define by lambda. <T> is dom object type that this method get property from it. We can achive the same effect by using another (old) interface.

### Syntax

```
CxQL
public CxList CxSelectDomProperty<T>(Func<T,IGraph> lambda) where T:CSharpGraph
```

**Parameters**

**lambda**

Method that define require DOM property.

**Return Value**

New list of requested properties.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

These examples demonstrates using of CxList.CxSelectDomProperty() method.
Example 1 : Get all TrueStatements of type IfStmt and Statements
of IterationStmt
//Current Solution
CxList False = Find_Always_False();
foreach (CxList t in False)
{
  CxList cond = t.GetFathers();
  if (cond.FindByType(typeof(IfStmt)).Count > 0)
  {
    IfStmt ifStmt = cond.data.GetByIndex(0) as IfStmt;
    falseBlocks.Add(ifStmt.TrueStatements.NodeId, ifStmt.TrueStatements);
  }
  else if (cond.FindByType(typeof(IterationStmt)).Count > 0)
  {
    IterationStmt iter = cond.data.GetByIndex(0) as IterationStmt;
    falseBlocks.Add(iter.Statements.NodeId, iter.Statements);
  }
}

// new solution
CxList False = Find_Always_False();
var cond = False.GetFathers();
var falseOfIf = cond.CxSelectDomProperty<IfStmt>(x => x.TrueStatements);
var falseOfIteration =
                 cond.CxSelectDomProperty<IterationStmt>(x => x.Statements);
var falseBlock = falseOfIf + falseOfIteration;

Example 2 :  Get some data based on “Left” property of AssignExpr
//Current Solution
foreach(CxList g in assignsExpr)
{
  AssignExpr ae = g.TryGetCSharpGraph<AssignExpr>();
  Expression e = ae.Left;
  CxList curNode = All.FindById(e.NodeId);
  left.Add(All.GetByAncs(curNode));
}
// new solution
CxList curNodes = assignsExpr.CxSelectDomProperty<AssignExpr>(x =>x.Left);
left.Add(All.GetByAncs(curNodes));
```

### Version Information

Supported from v9.2.0

## 5.12 CxList.CxSelectElements<T> Method (Func<T,IGraph, option) where T : CSharpGraph

Returns a new CxList of all required elements of input CxList. <T> is dom object type that this method get property from it. Main purpose of interface is hide internal DOM and CxList structures.

### Syntax

```
CxQL
```

```
public CxList CxSelectElements<T>(Func<T,IGraph> lambda, int option = -1)
where T:CSharpGraph
```

**Parameters**

**lambda**

Method that define require DOM property.

**option**

If option is -1 → iterate on all possible elements.

If option is 0 →return only first element.

**Return Value**

New CxList of all required elements of input CxList.

## Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

## Comments

The return value may be empty (Count = 0).

## Example

```
CxQL

These examples demonstrates using of CxList.CxSelectDomProperty() method.
Example 1 : Get first element of Indices of input CxList (query “Value_Shadowing”
C# medium)

//Current Solution
CxList variables = All.FindByType(typeof(IndexerRef));
CxList problematic = variables.FindByTypes(new string[]
{"Request","HttpRequest"});

foreach(KeyValuePair<int,IGraph> elem in problematic.data)
{
  try
  {
    IndexerRef ir = elem.Value as IndexerRef;
    CSharpGraph el = ir.Indices[0];
    result.Add(el.NodeId, el);
  }
  catch(Exception exc)
  {
    cxLog.WriteDebugMessage(exc);
  }
}

// new solution
CxList variables = All.FindByType(typeof(IndexerRef));
CxList problematic = variables.FindByTypes(new string[]
{"Request","HttpRequest"});
result = problematic.CxSelectElements<IndexerRef>(x=>x.Indices,0);

Example 2 :  query Find_Array_Indexes (GO, general)
//Current Solution
CxList arraysOrSlices = Find_IndexerRefs();
CxList arrayAccesses = All.NewCxList();
```

```
foreach(CxList arrayOrSlice in arraysOrSlices)
{
  IndexerRef idxRef = arrayOrSlice.TryGetCSharpGraph<IndexerRef>();
  foreach (var expr in idxRef.Indices)
  {
    if (expr is CSharpGraph)
    {
      arrayAccesses.Add(expr.NodeId, expr);
    }
  }
}
result = arrayAccesses;

// new solution
CxList arraysOrSlices = Find_IndexerRefs();
result = arraysOrSlices.CxSelectElements<IndexerRef>(x=>x.Indices);
```

### Version Information

Supported from v9.2.0

## 5.13 CxList.CxSelectElementsValues<T,I>(Func<T,I>) where: T : CSharpGraph

Returns list of requires property values of dom object of <TDomObject> and return list of <TOutput>. For more details see example. Main purpose of this method is hide internal CxList structure (data)

### Syntax

```
CxQL
public List<TOutput> CxSelectElementValues<TDomObject,TOutput> (Func<
TDomObject,TOutput > lambda) where TDomObject:CSharpGraph
```

**Parameters**

**lambda**

Method that define property to extract from require dom object.

**Return Value**

New List of all required values..

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

These examples demonstrates using of CxList.CxSelectElementValues() method.
Example : Compare parameter name of two methods (implementation and
declaration) If name of parameters are different add to result method declaration
and method implementation
```

```
(query “R16_04_Different_Identifiers_In_Function_Definition_And_Prototype” CPP
Misra)

//Current Solution
for (int i = 0;i < curParams.Count;i++)
{
   ParamDecl cur = curParams.curParams.data.GetByIndex(i) as ParamDecl;
   ParamDecl comp = compParams.data.GetByIndex(i) as ParamDecl;
   if (String.Compare(cur.Name, comp.Name) != 0)
   {
        result.Add(curMethodDecl + compMethodDecl);
        break;
   }
}

// new solution
var curListNames  =
       curParams.CxSelectElementValues<ParamDecl,string>(x => x.Name);
var compListNames =
       compParams.CxSelectElementValues<ParamDecl,string>(x => x.Name);
for (int i = 0; i < curListNames.Count; i++)
{
    if (String.Compare(curListNames[i], compListNames[i]) != 0)
    {
        result.Add(curMethodDecl + compMethodDecl);
        break;
    }
}
```

### Version Information

Supported from v9.2.0

## 5.14 CxList.ExtractFromSOQL Method ()

Extracts the parameters of a SOQL statement into a dictionary.

### Syntax

```
CxQL
public Dictionary<String, List<String>> ExtractFromSOQL()
```

**Return Value**

A dictionary with keys that match SOQL keywords and their relevant parameters.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.ExtractFromSOQL() method.
 The input source code is:
```

```
int b = 0;
String a = "select * from table where x=" + b;

Dictionary <String, List<String>> result = All.ExtractFromSOQL();

 the result would consist of 3 results:
             { "select" : "*",
              "from"   : "table",
              "where" : "x=" }
```

## 5.15 CxList.ExtractFromSOQL Method (string)

Extracts the parameters of the given keyword from a SOQL statement into a list.

### Syntax

```
CxQL
public List<string> ExtractFromSOQL(string keyword)
```

**Parameters**

**keyword**

The SOQL keyword to extract.

**Return Value**

A list with the parameters of the keyword.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the ExtractFromSOQL method.

```
CxQL

 This example demonstrates the CxList.ExtractFromSOQL() method.
 The input source code is:

int b = 0;
String a = "select * from table where x=" + b;

List <String> result = All.ExtractFromSOQL("select");

 the result would be -
      1 item found:
              ["*"]
```

## 5.16 CxList.DataInfluencedBy Method (CxList)

Returns a CxList which is a subset of "this" instance and its elements are data influenced by the CxList specified in parameter.

This call is equivalent to the following calls and it is recommended to use the short call format by default:

- DataInfluencedBy(list, InfluenceAlgorithmCalculation.OldAlgorithm)

### Syntax

```
CxQL
public CxList DataInfluencedBy(CxList influencing)
```

**Parameters**

**influencing**

CxList data-influencing on "this" instance.

**Return Value**

A subset of "this" instance data influenced by the specified CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.DataInfluencedBy() method.
 The input source code is:

 int b, a = 5;
 if (a > 3)
      b = a;

CxList five = All.FindByName("5");
result = All.DataInfluencedBy(five);

 the result would be -
      6 items found:
             a (in a = 5),
             a (in a > 3),
             > (in a > 3),
             a (in b = a),
             = (in b = a),
             b (in b = a)
```

## 5.17 CxList.DataInfluencedBy Method (CxList, InfluenceAlgorithmCalculation)

Returns a CxList which is a subset of this instance and its elements are data influenced by the CxList specified in the first parameter using the influence algorithm specified in the second parameter.

### Syntax

```
CxQL
public CxList DataInfluencedBy(CxList influencing, InfluenceAlgorithmCalculation
algorithm)
```

**Parameters**

**influencing**

CxList data-influencing on "this" instance.

**algorithm**

An enum matching the relevant InfluenceAlgorithmCalculation options which are:

OldAlgorithm, NewAlgorithm

**Return Value**

A subset of "this" instance data influenced by the specified CxList.

## Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

## Remarks

The return value may be empty (Count = 0).

## Example

```
CxQL

 This example demonstrates the CxList.DataInfluencedBy() method.
 The input source code is:
 int b, a = 5;
 if (a > 3)
      b = a;

CxList five = All.FindByName("5");
result = All.DataInfluencedBy(five,
CxList.InfluenceAlgorithmCalculation.NewAlgorithm);
 the result would be -
      6 items found:
             a (in a = 5),
             a (in a > 3),
             > (in a > 3),
             a (in b = a),
             = (in b = a),
             b (in b = a)
```

## 5.18 CxList.DataInfluencingOn Method (CxList)

Returns a CxList which is a subset of "this" instance and its elements are data influencing on the CxList specified in parameter.

This call is equivalent to the following calls and it is recommended to use the short call format by default:

- `DataInfluencingOn(list, InfluenceAlgorithmCalculation.OldAlgorithm)`

## Syntax

```
CxQL
public CxList DataInfluencingOn(CxList influenced)
```

**Parameters**

**influenced**

CxList data-influenced by "this" instance.

**Return Value**

A subset of "this" instance data influencing on the specified CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.DataInfluencingOn() method.
 The input source code is:

 int b, a = 5;
 if (a > 3)
      b = a;

CxList b = All.FindByName("*.b");
result = All.DataInfluencingOn(b);

 the result would be -
      3 items found:
             a (in b = a),
             a (in a = 5),
             5 (in a = 5)
```

## 5.19 CxList.DataInfluencingOn Method (CxList, InfluenceAlgorithmCalculation)

Returns a CxList which is a subset of "this" instance and its elements are data influencing on the CxList specified in the first parameter using the influence algorithm specified in the second parameter.

### Syntax

```
CxQL
public CxList DataInfluencingOn(CxList influenced, InfluenceAlgorithmCalculation
algorithm)
```

**Parameters**

**influenced**

CxList data-influenced by "this" instance.

**algorithm**

An enum matching the relevant InfluenceAlgorithmCalculation options which are:

OldAlgorithm, NewAlgorithm

**Return Value**

A subset of "this" instance data influencing on the specified CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.DataInfluencingOn() method.
 The input source code is:

 int b, a = 5;
 if (a > 3)
      b = a;

CxList b = All.FindByName("*.b");
result = All.DataInfluencingOn(b,
              CxList.InfluenceAlgorithmCalculation.NewAlgorithm);

 the result would be -
      3 items found:
             a (in b = a),
             a (in a = 5),
             5 (in a = 5)
```

## 5.20 CxList.InfluencedBy Method (CxList)

Returns a CxList which is a subset of "this" instance and its elements are influenced (either data or control) by the CxList specified in parameter.

This call is equivalent to the following calls and it is recommended to use the short call format by default:

- `InfluencedBy(list, InfluenceAlgorithmCalculation.OldAlgorithm)`

### Syntax

```
CxQL
public CxList InfluencedBy(CxList influencing)
```

**Parameters**

**influencing**

CxList data-influencing on "this" instance.

**Return Value**

A subset of "this" instance influenced by (either data or control) the specified CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.InfluencedBy() method.
 Notice the difference between DataInfluencedBy  and InfluencedBy
 The input source code is:
 int b = 2, a = 5, c;
 if (a > b)
      b = 3;
 c = b;
result = All.InfluencedBy(All.FindById(43)); // Id 43 is 5 from a = 5;
 Notice that among all the results also c (in c = b) appears because c is
 data-dependant on  b=3, which in turn is control dependant on a > b, which
itself is data-dependant on a = 5.

result = All.DataInfluencedBy(All.FindById(43)); // 5
 Notice that now c (in c = b) doesn't appear because its value is not
 influenced by 5.
```

## 5.21 CxList.InfluencedBy Method (CxList, InfluenceAlgorithmCalculation)

Returns a CxList which is a subset of "this" instance and its elements are influenced (either data or control) by the CxList specified in the first parameter using the influence algorithm specified in the second parameter.

### Syntax

```
CxQL
public CxList InfluencedBy(CxList influencing, InfluenceAlgorithmCalculation
algorithm)
```

**Parameters**

**influencing**

CxList data-influencing on "this" instance.

**algorithm**

An enum matching the relevant InfluenceAlgorithmCalculation options which are:

OldAlgorithm, NewAlgorithm

**Return Value**

A subset of "this" instance influenced by (either data or control) the specified CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.InfluencedBy() method.
 Notice the difference between DataInfluencedBy  and InfluencedBy
 The input source code is:
 int b = 2, a = 5, c;
 if (a > b)
      b = 3;
 c = b;
result = All.InfluencedBy(All.FindById(43),
       CxList.InfluenceAlgorithmCalculation.NewAlgorithm); // Id 43 is 5 from a
= 5;
```

```
 Notice that among all the results also c (in c = b) appears because c is
 data-dependant on  b = 3, which in turn is control dependant on a > b, which
itself is data-dependant on a = 5.

result = All.DataInfluencedBy(All.FindById(43)); // 5
 Notice that now c (in c = b) doesn't appear because its value is not
 influenced by 5.
```

## 5.22 CxList.InfluencedByAndNotSanitized Method (CxList, CxList)

Returns a CxList which is a subset of "this" instance and its elements are influenced by the CxList specified in the first parameter, and their influencing path doesn't contain elements from the CxList specified in the second parameter.

This call is equivalent to the following calls and it is recommended to use the short call format by default:

- `InfluencedByAndNotSanitized(influencing, sanitized, InfluenceAlgorithmCalculation.OldAlgorithm)`

### Syntax

```
CxQL
public CxList InfluencedByAndnotSanitized(CxList influencing, CxList
sanitization)
```

**Parameters**

**Influencing**

CxList influencing on "this" instance.

**sanitization**

CxList that "cuts" the influencing path.

**Return Value**

A subset of "this" instance and its elements are influenced by the first specified parameter, and their influencing path doesn't contain element from the second CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.InfluencedByAndNotSanitized()
 method.
 The input source code is:


           string s = input();
           string s1 = fixSql(s);
           string s2 = s + s1;

           execute(s);      (*)
           execute(s1);
```

```
            execute(s2);     (*)

            s = s1;
            execute(s);
            execute(s1);
            execute(s2);     (*)

            s2 = s;
            execute(s);
            execute(s1);
            execute(s2);

CxList execute = All.FindByName("execute");
CxList input = All.FindByName("input");
CxList fixSql = All.FindByName("fixSql");
result = execute.InfluencedByAndNotSanitized(input, fixSql);

 Notice that only the lines marked with a (*) are returned. These are the
 only statements that have an influencing path from the input() command,
 without being completely sanitized by fixSql().
```

## 5.23 CxList.InfluencedByAndNotSanitized Method (CxList, CxList, InfluenceAlgorithmCalculation)

Returns a CxList which is a subset of “this” instance and its elements are influenced by the CxList specified in the first parameter, and their influencing path doesn’t contain elements from the CxList specified in the second parameter, using the influence algorithm specified in the third parameter.

### Syntax

CxQL

```
public CxList InfluencedByAndnotSanitized(CxList influencing, CxList
sanitization, InfluenceAlgoritmCalculation algorithm)
```

**Parameters**

**influencing**

CxList influencing on “this” instance.

**sanitization**

CxList that "cuts" the influencing path.

**algorithm**

An enum matching the relevant InfluenceAlgorithmCalculation options which are:

OldAlgorithm, NewAlgorithm

**Return Value**

A subset of “this” instance and its elements are influenced by the first specified parameter, and their influencing path doesn’t contain element from the second CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.InfluencedByAndNotSanitized()
 method.
 The input source code is:

            string s = input();
            string s1 = fixSql(s);
            string s2 = s + s1;

            execute(s);      (*)
            execute(s1);
            execute(s2);     (*)

            s = s1;
            execute(s);
            execute(s1);
            execute(s2);     (*)

            s2 = s;
            execute(s);
            execute(s1);
            execute(s2);

CxList execute = All.FindByName("execute");
CxList input = All.FindByName("input");
CxList fixSql = All.FindByName("fixSql");
result = execute.InfluencedByAndNotSanitized(input, fixSql,
          CxList.InfluenceAlgorithmCalculation.NewAlgoritm);

 Notice that only the lines marked with a (*) are returned. These are the
 only statements that have an influencing path from the input() command,
 without being completely sanitized by fixSql().
```

## 5.24 CxList.InfluencingOn Method (CxList)

Returns a CxList which is a subset of “this” instance and its elements are influencing (data and control) on the CxList specified in parameter.

This call is equivalent to the following calls and it is recommended to use the short call format by default:

- `InfluencingOn(influenced, InfluenceAlgorithmCalculation.OldAlgorithm)`

### Syntax

```
CxQL
public CxList InfluencingOn (CxList influenced)
```

**Parameters**

**influenced**

CxList influenced by “this” instance.

**Return Value**

A subset of “this” instance influencing on the specified CxList.

### Exceptions

| Exception type | Condition |
|---|---|

| | |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.InfluencingOn() method.
 The input source code is:

 int a;
 a = 5;
 b = a;

CxList b_var = All.FindByShortName("b");
result = All.InfluencingOn(b_var);

 the result would be -
      3 items found:
             5,
             a,
              a
```

### Version Information

**CxAudit**

Supported from  v1.8.1

## 5.25 CxList.InfluencingOn Method (CxList, InfluenceAlgorithmCalculation)

Returns a CxList which is a subset of "this" instance and its elements are influencing (data and control) on the CxList specified in the first parameter using the influence algorithm specified in the second parameter.

### Syntax

```
CxQL
public CxList InfluencingOn (CxList influenced, InfluenceAlgorithmCalculation
algorithm)
```

**Parameters**

**influenced**

CxList influenced by "this" instance.

**algorithm**

An enum matching the relevant InfluenceAlgorithmCalculation options which are:

OldAlgorithm, NewAlgorithm

**Return Value**

A subset of "this" instance influencing on the specified CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.InfluencingOn() method.
 The input source code is:

 int a;
 a = 5;
 b = a;

CxList b_var = All.FindByShortName("b");
result = All.InfluencingOn(b_var,
                 CxList.InfluenceAlgorithmCalculation.NewAlgorithm);

 the result would be -
      3 items found:
             5,
             a,
             a
```

## 5.26 CxList.InfluencingOnAndNotSanitized Method (CxList,CxList)

Returns a CxList which is a subset of "this" instance and its elements are influencing on (Data or Control), and an influencing path exists which doesn't contain elements from the sanitization.

This call is equivalent to the following calls and it is recommended to use the short call format by default:

- `InfluencingOnAndNotSanitized(list, InfluenceAlgorithmCalculation.OldAlgorithm)`

### Syntax

```
CxQL
public CxList InfluencingOnAndNotSanitized (CxList influencing, CxList
sanitization)
```

**Parameters**

**influencing**

CxList influencing on "this" instance.

**sanitization**

CxList that "cuts" the influencing path

**Return Value**

A subset of "this" instance and its elements are influencing on the first specified parameter, and their influencing path doesn't contain elements from the CxList specified in second parameter.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.InfluencingOnandNotSanitized()
 method.
 The input source code is:

            string s = input();
            string s1 = fixSql(s);
            string s2 = s + s1;

            execute(s);      (*)
            execute(s1);
            execute(s2);     (*)

            s = s1;
            execute(s);
            execute(s1);
            execute(s2);     (*)

            s2 = s;
            execute(s);
            execute(s1);
            execute(s2);
CxList execute = All.FindByName("execute");
CxList input = All.FindByName("input");
CxList fixSql = All.FindByName("fixSql");
result = input.InfluencingOnAndNotSanitized(execute, fixSql);

 Notice that only the first line is returned (string s = input();)
```

## 5.27 CxList.InfluencingOnAndNotSanitized Method (CxList,CxList,InfluenceAlgorithmCalculation)

Returns a CxList which is a subset of “this” instance and its elements are influencing on (Data or Control), and an influencing path exists which doesn't contain elements from the sanitization using the influence algorithm specified in the third parameter.

### Syntax

```
CxQL
public CxList InfluencingOnAndNotSanitized (CxList influencing, CxList
sanitization, InfluenceAlgorithmCalculation algorithm)
```

**Parameters**

**influencing**

CxList influencing on this instance.

**sanitization**

CxList that "cuts" the influencing path

**algorithm**

An enum matching the relevant InfluenceAlgorithmCalculation options which are:

OldAlgorithm, NewAlgorithm

**Return Value**

A subset of "this" instance and its elements are influencing on the first specified parameter, and their influencing path doesn't contain elements from the CxList specified in the second parameter.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.InfluencingOnandNotSanitized()
 method.
 The input source code is:

           string s = input();
           string s1 = fixSql(s);
           string s2 = s + s1;

           execute(s);      (*)
           execute(s1);
           execute(s2);     (*)

           s = s1;
           execute(s);
           execute(s1);
           execute(s2);     (*)

           s2 = s;
           execute(s);
           execute(s1);
           execute(s2);
CxList execute = All.FindByName("execute");
CxList input = All.FindByName("input");
CxList fixSql = All.FindByName("fixSql");
result = input.InfluencingOnAndNotSanitized(execute, fixSql,
           CxList.InfluenceAlgorithmCalculation.NewAlgorithm);

 Notice that only the first line is returned (string s = input();)

 This query would return 1 result with a path from p declaration to the
Console.WriteLine, passing through the MemberAccess p.X
```

## 5.28 CxList.NotInfluencedBy Method (CxList)

Returns a CxList which is a subset of "this" instance and its elements are not influenced (either data or control) by the CxList specified in parameter.

### Syntax

```
CxQL
```

```
public CxList NotInfluencedBy(CxList influencing)
```

**Parameters**

**influencing**

CxList data on "this" instance.

**Return Value**

A subset of "this" instance not influenced by (either data or control) the specified CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.NotInfluencedBy() method.

 The input source code is:
 int b = 2, a = 5, c;
 if (a > b)
      b = 3;
 c = b;

result = All.NotInfluencedBy(All.FindById(43)); // 5
 Returns every object besides a and 5 (from a = 5) and a (from a > b)
 Notice that among all the results also b (in b = 2) appears because it does not
influence the value of a.
```

## 5.29 CxList.NotInfluencingOn Method (CxList)

Returns a CxList which is a subset of "this" instance and its elements are not influencing (data and control) on the CxList specified in parameter.

### Syntax

```
CxQL
public CxList NotInfluencingOn (CxList notInfluenced)
```

**Parameters**

**notInfluenced**

CxList data in "this" instance.

**Return Value**

A subset of "this" instance not influencing on the specified CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.NotInfluencingOn() method.
 The input source code is:

 int a, c = 3;
 a = 5;
 b = a;

CxList b_var = All.FindByShortName("b");
result = All.NotInfluencingOn(b_var);

 the result would be -
      2 items found:
             c,
             3
```

## 5.30 CxList Filter Methods

Here is definition of CSharpGraph dom object DTO (data transfare object)

```
public struct DOMProperties
{
   public string     ShortName; // dom object short name
   public string     FileName;  // file name of dom object
   public int        FileId;    // file id of dom object
   public string     FullName;  // dom object full name
   public GraphTypes GraphType; // represent type of dom object
   public int        Line;      // source line
   public int        Column;    // source column
}
```

### 5.30.1 CxList.GetDOMPropertiesOfFirst()

This method implements DTO of CSharpGraph dom object. It returns DTO of first dom object in CxList. Important reason for this interface is to hide internal structure of CSharpGraph dom object.

### Syntax

```
CxQL
public DOMProperties GetDOMProperties();
```

**Parameters**

**Return Value**

A DTO of first element of the given CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Version Information

Supported from v9.2.0

### 5.30.2 CxList.Filter(Func<DOMProperties,bool)

This method implements a new filter. It returns subset of "this". It is very similar to "Where" of LINQ.

#### Syntax

```
CxQL
public CxList Filter(Func<DOMProperties, bool> condition);
```

**Parameters**

**Condition**

Lambda method that define filter condition

**Return Value**

A subset of "this" instance, with elements that fulfilled lambda condition in the given CxList.

#### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

#### Comments

The return value may be empty (Count = 0).

#### Example

```
CxQL
This example demonstrates using of the CxList.Filter() method.
Get all Dom object with short name less than 50 characters

// current implementation
foreach (CxList r in tempResult)
{
   CSharpGraph g = r.data.GetByIndex(0) as CSharpGraph;
   If (g == null || g.ShortName == null)
   {
      continue;
   }
   if (g.ShortName.Length < 50)
   {
      result.Add(r);
   }
}

// new implementation
result = tempResult.Filter(x => x.ShortName.Length < 50);


This example demonstrates using of the CxList.Filter() and
GetDOMPropertiesOfFirst methods.

// current implementation
CxList sanitizers = All.NewCxList();
//Get all the elements that appear only after the position (line) of the header
foreach(CxList header in good_content_header_methods)
{
   CSharpGraph header_obj = header.TryGetCSharpGraph<CSharpGraph>();
   CxList methods_after_header =
           possible_sanitizers.GetByAncs(header.GetFathers().GetFathers());
   foreach(CxList method_after_header in methods_after_header)
   {
      CSharpGraph method_after_header_obj = method_after_header.
                                          TryGetCSharpGraph<CSharpGraph>();
      if(method_after_header_obj.LinePragma.Line >=
                                               header_obj.LinePragma.Line)
      {
         sanitizers.Add(method_after_header);
      }
   }
}

// new implementation
CxList sanitizers = All.NewCxList();
//Get all the elements that appear only after the position (line) of the header
foreach(CxList header in good_content_header_methods)
{
   int HeaderLineNumber = header.GetDOMPropertiesOfFirst().Line;
   CxList methods_after_header =
            possible_sanitizers.GetByAncs(header.GetFathers().GetFathers());
   sanitizers.Add(methods_after_header.Filter(x => x.Line >=
                                                         HeaderLineNumber));}
```

### Version Information

Supported from v9.2.0

## 5.30.3 CxList.FilterByDomProperty<T> Method (Func<T, bool>) where T:CSharpGraph

Returns a CxList which is a subset of "this" instance, with elements that match input lambda method

### Syntax

```
CxQL
public CxList FilterByDomProperty<T>(Func<T, bool> condition) where T:
CSharpGraph;
```

**Parameters**

**condition**

Lambda method that define/implement require condition.

**Return Value**

A subset of "this" instance, with elements that fulfilled lambda condition in the given CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL
```

```
This example demonstrates using of the CxList.FilterByDomProperty() method. Get
all AssignExpr with operator equal to AdditionAssign

// current implementation
foreach(CxList assignment in assignments)
{
    AssignExpr graph = assignment.TryGetCSharpGraph<AssignExpr>();
    if (graph != null && graph.Operator == AssignOperator.AdditionAssign)
       assignAdd.Add(assignment);
}

// new implementation
CxList assignAdd = assignments.
              FilterByDomProperty<AssignExpr>(x =>x.Operator ==
                                                AssignOperator.AdditionAssign);
```

### Version Information

Supported from v9.2.0

### 5.30.4 CxList.FilterPlugins Method ()

Returns a CxList which is a subset of "this" instance, with elements that are objects from CxEngine Plugins removed.

### Syntax

```
CxQL
public CxList FilterPlugins(void)
```

**Return Value**

A subset of "this" instance, with elements that are declared in the CxEngine plugins removed.

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FilterPlugins() method.
 Consider processing C code

#include <stdlib.h>

Void main(void) {
 Void *ptr = malloc(sizeof(int));
 return 0;
}

CxList mallocDef = All.FindByType<MethodDecl>().FindByName("malloc");
// this list contains one node, the definition of malloc in stdlib.h
CxList localMallocDef = mallocDef.FilterPlugin();
// this list would be empty
```

### Version Information

Supported from v9.5.1

## 5.31 CxList.FindAllMembers Method (CxList)

Returns a CxList which is a subset of "this" instance, with elements that are members of the classes in the given CxList.

### Syntax

```
CxQL
public CxList FindAllMembers(CxList Ids)
```

**Parameters**

**Ids**

The list of Classes whose members are to be found.

**Return Value**

A subset of “this” instance, with elements that are members of the classes in the given CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindAllMembers() method.
 The input source code is:

public class MyClass
{
 public int b, a = 5;
 boolean c=false;
}
result = All.FindAllMembers(All.FindByName(“MyClass”));
 The result would consist of 3 items:
            b (public int b, a = 5;),
            a (public int b, a = 5;),
            c (boolean c=false)
```

## 5.32 CxList.FindAllReferences Method (CxList)

Returns a CxList which is a subset of “this” instance, with elements that are references of the given CxList.

### Syntax

```
CxQL
public CxList FindAllReferences(CxList referenced)
```

**Parameters**

**referenced**

The CxList whose references are to be found.

**Return Value**

A subset of “this” instance, with elements that are references of the given CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL
```

```
 This example demonstrates the CxList.FindAllRefernces() method.
 The input source code is:

 int b, a = 5;
 if (a > 3)
      b = a;

result = All.FindAllReferences(All.FindById(36)); //a in (a = 5)

 the result would consist of 3 items:
            a (in a = 5),
            a (in a > 5),
            a (in b = a)
```

## 5.33 CxList.FindAllReferences Method (CxList, CxList)

Returns a CxList which is a subset of "this" instance, with elements that are references of the given CxList, excluding elements in the second CxList.

### Syntax

```
CxQL
public CxList FindAllReferences(CxList referenced, CxList exclude)
```

**Parameters**

**referenced**

The CxList whose references are to be found.

**exclude**

The CxList whose elements will be ignored and excluded.

**Return Value**

A subset of "this" instance, with elements that are references of the given CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindAllRefernces() method.
 The input source code is:

 int b, a = 5;
 if (a > 3)
      b = a;

result = All.FindAllReferences(All.FindById(36), All.FindById(30)); //a in (a =
5), b in (int b)

 the result would consist of 3 items:
```

```
             a (in a = 5),
             a (in a > 5),
             a (in b = a)
```

## 5.34 CxList.FindByAssignmentSide Method (AssignmentSide)

Returns a CxList which is a subset of "this" instance and its elements are being on the given side of an assignment expression.

### Syntax

```
CxQL
public CxList FindByAssignmentSide(CxList.AssignmentSide side)
```

**Parameters**

**side**

The side of the assignment expression, which can be one of the following values: Left, Right (see Section AssignmentSide).

**Return Value**

A subset of "this" instance on the specified side of an assignment expression.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByAssignmentSide() method.
 The input source code is:

 a = 3;
 b = a;
 if (a == 4)
      b = a - 1;

result = All.FindByAssignmentSide(CxList.AssignmentSide.Left);

 The result would consist of 3 items:
             a (in a = 3),
             b (in b = a),
             b (in b = a - 1)
```

### Version Information

Supported from v1.8.1

## 5.35 CxList.FindByCustomAttribute Method (string)

Returns a CxList which is a subset of "this" instance and its elements are custom attributes of the specified name.

### Syntax

```
CxQL
public CxList FindByCustomAttribute(string name)
```

**Parameters**

**name**

The attribute name.

**Return Value**

A subset of "this" instance with custom attributes of the specified name.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByCustomAttribute() method.
 The input source code is:

 [WebMethod]
 void foo()
 {

 }

result = All. FindByCustomAttribute("WebMethod");

 the result would consist of 1 item:
            foo
```

## 5.36 CxList.FindByExtendedType Method (string)

Returns a CxList which is a subset of "this" instance and the type of its elements match the type specified as parameter.

### Syntax

```
CxQL
public CxList FindByExtendedType (string extendedType)
```

**Parameters**

**extendedType**

The extended type of the objects to be found. Prefix and postfix wildcard (*) are supported.

**Return Value**

A subset of "this" instance and its elements are those with type specified by the parameter.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByExtendedType() method.
 The input source code is:

MyClass a;
 MyClassExtended b;
 int c;
 a.DataMember = 3;
 c = a.Method();

result = All.FindByExtendedType ("MyClass*");
 the result would consists of 4 items:
             a (in MyClass a)
             b (in MyClassExtended b)
             a (in a.DataMember = 3)
             a (in c = a.Method())
```

## 5.37 CxList.FindByFathers Method (CxList)

Returns a CxList which is a subset of “this” instance and its elements are those that their CxDOM-Fathers are in the specified CxList.

### Syntax

```
CxQL
public CxList FindByFathers(CxList fathers)
```

**Parameters**

**fathers**

A CxList consisting of the Fathers to be matched.

**Return Value**

A subset of “this” instance and its elements are those which their CxDOM-Fathers are in the specified CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL
 This example demonstrates the CxList.FindByFathers() method.
 First we find the number "3", then we seek for 3's fathers (which are
 the assignment expressions), finally we look for the
 assignment-expressions' sons (the "a", "b" and the "3"s)
 Input source code is:

 a = 3;
 b = a;
 if (a == 4)
      b = a - 1;
CxList three = All.FindByName("*.3");
CxList threesFathers = three.GetFathers();
Result = All.FindByFathers(threesFathers);
 the result would be -
      4 items found:
             a (in a = 3),
             3 (in a = 3),
             b (in b = 3),
             3 (in b = 3)
```

## 5.38 CxList.FindByFieldAttributes Method (Modifiers)

Returns a CxList which is a subset of "this" instance and its elements are modified by the modifier (private, external, etc).

### Syntax

```
CxQL
public CxList FindByFieldAttributes(Modifiers attrib)
```

**Parameters**

**Attrib**

Attribute of the fields to be found.

**Return Value**

A subset of "this" instance and its elements are those with attribute attrib.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByAttributes() method.
 Input source code is:
 public class cl1{
          private void foo(){}
          protected void guu(){}
          private int a,b;
          protected int c;
  }

result=All.FindByFieldAttributes(Modifiers.Protected);
 the result would be -
      2 items found:
             guu (in protected void guu(){}),
              c (int c;)
```

### Version Information

Supported from **CxAudit** v2.0.5

## 5.39 CxList.FindByFileName Method (string)

Returns a CxList which is a subset of “this” instance and its elements are in a given source code file.

### Syntax

```
CxQL
public CxList FindByFileName(string FileName)
```

**Parameters**

**FileName**

String with the file name.

**Return Value**

A subset of "this” instance with elements from a given file name.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByFileName() method.

 The input source code is:

 //file myCode.cs
 class Cl {
       void foo() {
              int i;
       }
 }

result = All.FindByFileName("*myCode.cs");

 the result consists of 5 items:
             Cl
             void,
             foo,
               int,
               i
```

### Version Information

Supported from v1.8.1

## 5.40 CxList.FindById Method (int)

Finds all objects with the specified id. This method is mainly used to find all the uses of a code element (e.g. variable, class).

### Syntax

```
CxQL
public CxList FindById (int id)
```

**Parameters**

**id**

id number to be found.

**Return Value**

A subset of “this” instance and its elements that have the specified id number.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindById() method.
 The input source code is:

 a = 3;
 b = a;
 if (a == 4)
      b = a - 1;

result = All.FindById(60);

 the result would be -
      1 item found:
             b (in b = a - 1)
```

## 5.41 CxList.FindByInitialization Method (CxList)

Returns a CxList which is a subset of “this” instance and contains elements initialized by the given CxList.

### Syntax

```
CxQL
public CxList FindByInitialization(CxList initializators)
```

**Parameters**

**initializators**

A CxList with initializers to search in “this” instance.

**Return Value**

A subset of “this” instance containing declarators initialized by the specified CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByInitialization() method.
 The input source code is:

 int b = 5;

CxList declarators = All.FindByType(typeof(Declarator));
result= declarators.FindByInitialization(All);

 the result would consist of 1 item:
            b
```

### Version Information

Supported from v1.8.1

## 5.42 CxList.FindByLanguage Method (string)

Returns a CxList which is a subset of "this" instance whose elements are from the given language.

### Syntax

```
CxQL
public CxList FindByLanguage (string languageName)
```

**Parameters**

**languageName**

Language name to search.

**Return Value**

A subset of "this" instance whose elements are from the given language.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByLanguage() method.
 The input source code is:
 //file myCode.cs
 class myCode {
 }

 //file MyCode.java
 class MyCode {

 }

result = All.FindByLanguage ("Java");
 the result would consist of 1 item:
             myCode (class MyCode)
```

## 5.43 CxList.FindByMemberAccess Method (string)

Receives a qualified notation string (e.g. “T.M”) where T is a type name and M is a member name.

Returns a CxList which is a subset of “this” instance containing the elements with name equal to M and belong to a type whose name **ends with** T. This is a case sensitive search.

For a non case-sensitive search, use the `FindByMemberAccess Method (string, bool)` instead.

For a search by **exact** target type name, use `FindByExactMemberAccess(string)` instead.

### Syntax

```
CxQL
public CxList FindByMemberAccess(string memberAccess)
```

**Parameters**

**memberAccess**

Contains both the name of the type and the name of the accessed member in the qualified notation (eg. "CheckBoxList.SelectedValue"). Prefix and suffix wild card (*) are permitted.

**Return Value**

A subset of “this” instance where its elements are the ones which their given member is accessed.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByMemberAccess() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
result = All.FindByMemberAccess("Class.DataMember");
 the result would consist of 1 item:
             a.DataMember (in a.DataMember = 3)
This is so, because MyClass ends with Class.
```

```
result = All.FindByMemberAccess("MyClass.Met*");

 the result would consist of 1 item:
            a.Method (in b = a.Method())
```

## 5.44 CxList.FindByMemberAccess Method (string,bool)

Receives a qualified notation string (e.g. “T.M”) where T is a type name and M is a member name.

Returns a CxList which is a subset of “this” instance containing the elements with name equal to M and belong to a type whose name **ends with** T. This search allows both case-sensitive and non case-sensitive searches.

For a search by **exact** target type name, use `FindByExactMemberAccess(string)` instead.

### Syntax

```
CxQL
public CxList FindByMemberAccess(string memberAccess, bool caseSensitive)
```

**Parameters**

**memberAccess**

Contains both the name of the type and the name of the accessed member in qualified notation (eg. "CheckBoxList.SelectedValue"). Prefix and suffix wild card (*) are permitted.

**caseSensitive**

Boolean which indicates to the search to be (or not) case sensitive.

**Return Value**

A subset of “this” instance where its elements are the ones which their specified member is accessed.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByMemberAccess() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
result = All.FindByMemberAccess("MyClass.dataMember", true);
 Notice that the result would consist of 0 items because the search is case-
sensitive.

result = All.FindByMemberAccess("Class.dataMember", false);
 The result would consist of 1 item:
            a.DataMember (in a.DataMember = 3)
 This is so, because MyClass ends with Class.
```

```
result = All.FindByMemberAccess("MyClass.met*", true);
 Notice that the result would consist of 0 items because the search is case-
sensitive.


result = All.FindByMemberAccess("MyClass.met*", false);
 the result would consist of 1 item:
           a.Method (in b = a.Method())
```

### Version Information

Supported from v1.8.1

## 5.45 CxList.FindByMemberAccess Method (string,string)

Returns a CxList which is a subset of "this" instance containing the elements with name equal to the string in second parameter and belong to a type whose name **ends with** the string in the first parameter. This is a case-sensitive search by both parameters.

For a non case-sensitive search, use the `FindByMemberAccess Method(string, string, bool)` instead.

### Syntax

CxQL
```
public CxList FindByMemberAccess(string typeName, string memberName)
```

**Parameters**

**typeName**

Contains the name of the accessed type (eg. "CheckBoxList");

**memberName**

Contains the name of the accessed member (eg. "SelectedValue");

**Return Value**

A subset of "this" instance where its elements are the ones which their specified member is accessed.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

CxQL
```

 This example demonstrates the CxList.FindByMemberAccess() method.
 The input source code is:
 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
result = All.FindByMemberAccess("Class", "DataMember");

 The result would consist of 1 item:
```

```
             a.DataMember (in a.DataMember = 3)
 This is so, because MyClass ends with Class.

result = All.FindByMemberAccess("MyClass", "Met*");

 The result would consist of 1 item:
             a.Method (in b = a.DataMethod())
```

### Version Information

Supported from v7.9.0

## 5.46 CxList.FindByMemberAccess Method (string, string, bool)

Returns a CxList which is a subset of “this” instance containing the elements with name equal to the string in second parameter and belong to a type whose name **ends with** the string in the first parameter. This search allows both case-sensitive and non case-sensitive searches by the given parameters.

### Syntax

```
CxQL
public CxList FindByMemberAccess(string typeName, string memberName, bool
caseSensitive)
```

**Parameters**

**typeName**

Contains the name of the accessed type (eg. "CheckBoxList");

**memberName**

Contains the name of the accessed member (eg. "SelectedValue");

**caseSensitive**

Boolean which indicates to the search to be (or not) case sensitive.

**Return Value**

A subset of “this” instance where its elements are the ones which their specified member is accessed.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByMemberAccess() method.
 The input source code is:
 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
result = All.FindByMemberAccess("MyClass", "dataMember", true);

 the result would consist of 0 item because it is a case-sensitive search.
```

```
result = All.FindByMemberAccess("Class", "dataMember", false);

 the result would consist of 1 item:
            a.DataMember (in a.DataMember = 3)
 This is so, because MyClass ends with Class.


result = All.FindByMemberAccess("MyClass", "met*", true);



the result would consist of 0 item, because it is a case-sensitive search.

result = All.FindByMemberAccess("MyClass", "met*", false);

the result would consist of 1 item:
             a.Method (in b = a.DataMethod())
```

## 5.47 CxList.FindByMemberAccesses Method (string[],bool)

Receives an array of qualified notation strings (e.g. “T.M”) where T is a type name and M is a member name.

Returns a CxList which is a subset of “this” instance containing the elements with name equal to M and belong to a type whose name ends with T, for any strings in the array. This search allows both case-sensitive and non case-sensitive searches.

### Syntax

```
CxQL
public CxList FindByMemberAccesses(string [] memberAccesses, bool caseSensitive =
true)
```

**Parameters**

**memberAccesses**

An array of strings where each contains both the name of the type and the name of the accessed member in qualified notation
(eg. "CheckBoxList.SelectedValue"). Prefix and suffix wild card (*) are permitted.

**caseSensitive**

Boolean which indicates to the search to be (or not) case sensitive. This Boolean is true by default

**Return Value**

A subset of “this” instance where its elements are the ones which their specified members are accessed.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL
```

```
 This example demonstrates the CxList.FindByMemberAccesses() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
string []memberAcesses = new string[]{"Class.dataMember", "MyClass.Met*"}
result = All.FindByMemberAccesses(memberAcesses, true);
 Notice that the result would consist of 1 item because the search is case-
sensitive: a.Method()

result = All.FindByMemberAccesses(memberAcesses);
 The result would consist of 2 items:
       a.DataMember
       a.Method
 This is so, because MyClass ends with Class.
```

### Version Information

Supported from 9.0.0

## 5.48 CxList.FindByExactMemberAccess Method (string)

Receives a qualified notation string (e.g. "T.M") where T is a type name and M is a member name.

Returns a CxList which is a subset of "this" instance containing the elements with name equal to M and belong to a type whose name **equals** T. This is a case sensitive search.

For a non case-sensitive search, use the `FindByExactMemberAccess (string, bool)` instead.

### Syntax

```
CxQL
public CxList FindByExactMemberAccess(string memberAccess)
```

**Parameters**

**memberAccess**

Contains both the name of the type and the name of the accessed member in the qualified notation (eg. "CheckBoxList.SelectedValue").

**Return Value**

A subset of "this" instance where its elements are the ones which their given member is accessed.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByMemberAccess() method.
 The input source code is:
```

```
 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
result = All.FindByExactMemberAccess("MyClass.DataMember");
 the result would consist of 1 item:
             a.DataMember (in a.DataMember = 3)

result = All.FindByExactMemberAccess("Class.DataMember");
 the result would consist of 0 items, because “Class” is not equal to “MyClass”.
```

### Version Information

Supported from v8.2.0

## 5.49 CxList.FindByExactMemberAccess Method (string,bool)

Receives a qualified notation string (e.g. “T.M”) where T is a type name and M is a member name.

Returns a CxList which is a subset of “this” instance containing the elements with name equal to M and belong to a type whose name equals T. This search allows both case-sensitive and non case-sensitive searches.

### Syntax

```
CxQL
public CxList FindByExactMemberAccess(string memberAccess, bool caseSensitive)
```

**Parameters**

**memberAccess**

Contains both the name of the type and the name of the accessed member in qualified notation (eg. "CheckBoxList.SelectedValue").

**caseSensitive**

Boolean which indicates to the search to be (or not) case sensitive.

**Return Value**

A subset of “this” instance where its elements are the ones which their specified member is accessed.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByMemberAccess() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
```

```
result = All.FindByExactMemberAccess("MyClass.dataMember", true);
 Notice that the result would consist of 0 items because the search is case-
sensitive.

result = All.FindByExactMemberAccess("MyClass.dataMember", false);
 The result would consist of 1 item:
            a.DataMember (in a.DataMember = 3)

result = All.FindByExactMemberAccess("Class.dataMember", false);
The result would consist of 0 items, because “Class” is not equals to “MyClass”.
```

### Version Information

Supported from v8.2.0

## 5.50 CxList.FindByExactMemberAccess Method (string,string)

Returns a CxList which is a subset of “this” instance containing the elements with name equal to the string in second parameter belong to a type whose name equals the string in the first parameter. This is a case-sensitive search by parameters.

For a non case-sensitive search, use the `FindByExactMemberAccess Method(string, string, bool)` instead.

### Syntax

```
CxQL
public CxList FindByExactMemberAccess(string typeName, string memberName)
```

**Parameters**

**typeName**

Contains the name of the accessed type (eg. "CheckBoxList");

**memberName**

Contains the name of the accessed member (eg. "SelectedValue");

**Return Value**

A subset of “this” instance where its elements are the ones which their specified member is accessed.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByMemberAccess() method.
 The input source code is:
 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
result = All.FindByExactMemberAccess("MyClass", "DataMember");
```

```
 The result would consist of 1 item:
             a.DataMember (in a.DataMember = 3)

result = All.FindByExactMemberAccess("Class", "DataMember");
The result would consist of 0 items, because “Class” is not equal to “MyClass”.
```

### Version Information

Supported from v8.2.0

## 5.51 CxList.FindByExactMemberAccess Method (string, string, bool)

Returns a CxList which is a subset of “this” instance containing the elements with name equal to the string in second parameter and belong to a type whose name **equals** the string in the first parameter. This search allows both case-sensitive and non case-sensitive searches by both parameters.

### Syntax

```
CxQL
public CxList FindByExactMemberAccess(string typeName, string memberName, bool
caseSensitive)
```

**Parameters**

**typeName**

Contains the name of the accessed type (eg. "CheckBoxList");

**memberName**

Contains the name of the accessed member (eg. "SelectedValue");

**caseSensitive**

Boolean which indicates to the search to be (or not) case sensitive.

**Return Value**

A subset of “this” instance where its elements are the ones which their specified member is accessed.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByMemberAccess() method.
 The input source code is:
 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
result = All.FindByExactMemberAccess("MyClass", "dataMember", true);
  the result would consist of 0 item because it is a case-sensitive search.

result = All.FindByExactMemberAccess("MyClass", "dataMember", false);
 the result would consist of 1 item:
             a.DataMember (in a.DataMember = 3)
```

```
result = All.FindByExactMemberAccess("Class", "dataMember", false);
the result would consist of 0 items, because “Class”is not equal to “Class”.
```

### Version Information

Supported from v8.2.0

## 5.52 CxList.FindByExactoMemberAccesses Method (string[],bool)

Receives an array of qualified notation strings (e.g. “T.M”) where T is a type name and M is a member name.

Returns a CxList which is a subset of “this” instance containing the elements with name equal to M and belong to a type whose **name equals T**, for any strings in the array. This search allows both case-sensitive and non case-sensitive searches.

### Syntax

```
CxQL
public CxList FindByExactMemberAccesses(string [] memberAccesses, bool
caseSensitive = true)
```

**Parameters**

**memberAccesses**

An array of strings where each contains both the name of the type and the name of the accessed member in qualified notation
(eg. "CheckBoxList.SelectedValue"). Prefix and suffix wild card (*) are permitted.

**caseSensitive**

Boolean which indicates to the search to be (or not) case sensitive. This Boolean is true by default

**Return Value**

A subset of “this” instance where its elements are the ones which their specified members are accessed.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByExactMemberAccesses() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
string []memberAcesses = new string[]{“Class.dataMember”, “MyClass.Met*”}
result = All.FindByMemberAccesses(memberAcesses, false);
 Notice that the result would consist of 1 item because the search is exact:
a.Method()
```

### Version Information

Supported from 9.6.0

## 5.53 CxList.FindByMethodReturnType Method (string)

Returns a CxList which is a subset of "this" instance and its elements are method declarators of a given return type.

### Syntax

```
CxQL
public CxList FindByMethodReturnType(string type)
```

**Parameters**

**type**

The return type name string.

**Return Value**

A subset of "this" instance with method declarators of a given return type.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByMethodReturnType() method.
 The input source code is:

 MyType foo() {
       ...
 }

result = All.FindByMethodReturnType("MyType");

 the result would consists of 1 item:
           foo
```

## 5.54 CxList.FindByName Method (string)

Returns a CxList which is a subset of "this" instance and its elements are the ones which their name is the given parameter.

### Syntax

```
CxQL
public CxList FindByName(string name)
```

**Parameters**

**name**

The name of the objects to look for. Prefix and postfix wildcard (*) are supported.

**Return Value**

A subset of “this” instance and its elements are the ones which their name is the given parameter.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByName() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();

result = All.FindByName("*Member*");

 the result would consist of 1 item:
             a.DataMember (in a.DataMember = 3)
```

## 5.55 CxList.FindByName Method (string, int, int)

Returns a CxList which is a subset of “this” instance and its elements are the ones which their name is the given parameter (optionally with wildcards) and is not shorter than minLength and not longer than maxLength.

### Syntax

```
CxQL
public CxList FindByName(string name, int minLength, int maxLength)
```

**Parameters**

**name**

Contains the name of the objects. Prefix and postfix wildcard (*) are supported.

**minLength**

Minimum length of the searched strings.

**maxLength**

Maximum length of the searched strings.

**Result**

A subset of “this” instance and its elements are the ones which their name is the given parameter, according to the given length interval.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByName() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();

result = All.FindByName("*Me*",3,7);
 the result would consist of 1 item:
       Method (in b = a.Method())
```

### Version Information

Supported from v2.0.5

## 5.56 CxList.FindByName Method (string, bool)

Returns a CxList which is a subset of "this" instance and its elements are the ones which their name is the given parameter, according to the specified comparison criteria.

### Syntax

```
CxQL
public CxList FindByName(string name, bool caseSensitive)
```

**Parameters**

**name**

Contains the name of the objects. Prefix and postfix wildcard (*) are supported.

**caseSensitive**

Boolean which indicates to the search to be (or not) case sensitive.

**Return Value**

A subset of "this" instance and its elements are the ones which their name is the given parameter, according to the given comparison criteria. The caseSensitive boolean value defines the ability to search using case sensitive or case insensitive comparison.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByName() method.
 The input source code is:
```

```
 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();

result = All.FindByName("*member*",true);
 the result would consist of 0 items.

result = All.FindByName("*member*", false);
 the result would consist of 1 item:
       a.DataMember (in a.DataMember = 3)
```

### Version Information

Supported from v1.8.1

## 5.57 CxList.FindByName Method (string, StringComparison)

Returns a CxList which is a subset of "this" instance and its elements are the ones which their name is the given parameter. The comparison method specified in parameter is used for matching.

### Syntax

```
CxQL
public CxList FindByName(string name, StringComparison comparisonType)
```

**Parameters**

**name**

The name of the objects to look for. Prefix and postfix wildcard (*) are supported.

**comparisonType**

StringComparison type to be used in name comparison. One of the following values:

*CurrentCulture, CurrentCultureIgnoreCase, InvariantCulture, InvariantCultureIgnoreCase, Ordinal, OrdinalIgnoreCase*

**Return Value**

A subset of "this" instance and its elements are the ones which their name is the given parameter.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByName() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
```

```
result = All.FindByName("*member*", StringComparison.OrdinalIgnoreCase);
 the result would consist of 1 item:
       DataMember (in a.DataMember = 3)
```

## 5.58 CxList.FindByNames Method (CxList)

Returns a CxList which is a subset of “this” instance and its elements are the ones which their names are equal to the given list.

### Syntax

```
CxQL
public CxList FindByNames(CxList nodesList)
```

**Parameters**

**nodesList**

The list of nodes containing the names to be found.

**Return Value**

A subset of “this” instance and its elements are the ones which the name is contained in the given list.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL
 This example demonstrates the CxList.FindByName() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();

 result = All.FindByNames(All.FindByType(typeof(MemberAccess)));

 the result would consist of 3 items:
a (in MyClass a)
a (in a.DataMember = 3)
a (in b = a.Method())
```

## 5.59 CxList.FindByNames Method (CxList, bool)

Returns a CxList which is a subset of “this” instance and its elements are the ones which their names are equal to the list given.

### Syntax

```
CxQL
public CxList FindByNames(CxList nodesList, bool CaseSensitive)
```

**Parameters**

**nodesList**

The list of nodes containing the names to be found.

**CaseSensitive**

Boolean which indicates to the search to be (or not) case sensitive.

**Return Value**

A subset of “this” instance and its elements are the ones which their name is contained in the given list, according to the specified case sensitivity comparison criteria.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL
 This example demonstrates the CxList.FindByName() method.
 The input source code is:

 MyClass A;
 int a;
 A.DataMember = 3;
 a = A.Method();

result = All.FindByNames(All.FindByType(typeof(MemberAccess)), false);

 the result would consist of 5 items:
            A (in MyClass A)
            a (in int a)
            A (in A.DataMember = 3)
            a (in a = A.Method())
            A (in a = A.Method())
```

## 5.60 CxList.FindByNames Method (param string[])

Returns a CxList which is a subset of “this” instance and its elements are the ones which their names are equal to the list given.

### Syntax

```
CxQL
public CxList FindByNames(param string[] nodesList)
```

**Parameters**

**nodesList**

A list of strings containing the names to be found.

**Return Value**

A subset of “this” instance and its elements are the ones which their name is contained in the given list, according to the specified case sensitivity comparison criteria.

CaseSensitive is by default is true.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL
 This example demonstrates the CxList.FindByName() method.
 The input source code is:

 MyClass A;
 int a;
 A.DataMember = 3;
 a = A.Method();

result = All.FindByNames("*MEMBER","A.Method");

 the result would consist of 3 items:
a (in a.DataMember = 3)
a (in b = a.Method())
a (in b = a.Method())
```

### Version Information

Supported from v9.4.0

## 5.61 CxList.FindByNames Method (string[], bool)

Returns a CxList which is a subset of “this” instance and its elements are the ones which their names are equal to the list given.

### Syntax

```
CxQL
public CxList FindByNames(string[] nodesList, bool CaseSensitive = true)
```

**Parameters**

**nodesList**

A list of strings containing the names to be found.

**CaseSensitive**

Boolean which indicates to the search to be (or not) case sensitive.

**Return Value**

A subset of “this” instance and its elements are the ones which their name is contained in the given list, according to the specified case sensitivity comparison criteria.

CaseSensitive is by default is true.

### Exceptions

| Exception type | Condition |
|---|---|

| | |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

```
Example
CxQL
 This example demonstrates the CxList.FindByName() method.
 The input source code is:

 MyClass A;
 int a;
 A.DataMember = 3;
 a = A.Method();
string[] str = new string[2] {"*MEMBER","A.Method"};

result = All.FindByNames(str);

 the result would consist of 3 items:
a (in a.DataMember = 3)
a (in b = a.Method())
a (in b = a.Method())
```

### Version Information

Supported from v9.4.0

## 5.62 CxList.FindByNames Method (string[], StringComparison)

Returns a CxList which is a subset of "this" instance and its elements are the ones which their names are equal to the list given.

### Syntax

```
CxQL
public CxList FindByNames(string[] nodesList, StringComparison comparisonType)
```

**Parameters**

**nodesList**

A list of strings containing the names to be found.

**comparisonType**

StringComparison type to be used in name comparison. One of the following values:

*CurrentCulture, CurrentCultureIgnoreCase, InvariantCulture, InvariantCultureIgnoreCase, Ordinal, OrdinalIgnoreCase*

**Return Value**

A subset of "this" instance and its elements are the ones which their name is contained in the given list, according to the specified case sensitivity comparison criteria.

StringComparison is by default is *Ordinal*.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL
 This example demonstrates the CxList.FindByName() method.
 The input source code is:

 MyClass A;
 int a;
 A.DataMember = 3;
 a = A.Method();
 string[] str = new string[2] {"*MEMBER","A.Method"};

result = All.FindByNames(str, StringComparison.OrdinalIgnoreCase);

 the result would consist of 3 items:
a (in a.DataMember = 3)
a (in b = a.Method())
a (in b = a.Method())
```

### Version Information

Supported from v9.4.0

## 5.63 CxList.FindByParameterName(string)

Returns a CxList which is a subset of "this", containing only MethodInvokeExpr DOM node where their arguments are labeled according to the given value.

### Syntax

```
CxQL
public CxList FindByParameterName (string paramName)
```

**Parameters**

**paramName**

String containing the name of the parameter belonging to the resultant methods.

**Return Value**

A subset of "this" instance where its elements are MethodInvokeExpr and ObjectCreateExpr and these contain arguments labelled according to 'paramName'.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList. FindByParameterName(string) method.
 The input source code is:
```

```
 class NamedExample
{
    static void Main(string[] args)
    {
        PrintOrderDetails(sellerName:"Gift Shop",31,productName:"Red Mug");
    }
}

result = All.FindByParameterName (“sellerName”);

 the result would consist in 1 item:
             PrintOrderDetails
```

## 5.64 CxList.FindByParameterName(string, int)

Returns a CxList which is a subset of “this”, containing only MethodInvokeExpr DOM node where their arguments on a given position are labeled according to the given value.

### Syntax

CxQL
```
public CxList FindByParameterName (string paramName, int paramPosition)
```

**Parameters**

**paramName**

String containing the name of the parameter belonging to the resultant methods.

**paramPosition**

Zero based index indicating the position of argument named ‘paramName’.

**Return Value**

A subset of “this” instance where its elements are MethodInvokeExpr and ObjectCreateExpr and these contain arguments labelled according to ‘paramName’ in the indicated position by ‘paramPosition’.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

CxQL
```
 This example demonstrates the CxList. FindByParameterName(string, int) method.
 The input source code is:

 class NamedExample
{
    static void Main(string[] args)
    {
        PrintOrderDetails(sellerName:"Gift Shop",31,productName:"Red Mug");
    }
}

result = All.FindByParameterName (“sellerName”, 1);
```

```
 would have no results because there’s no argument named “sellerName” on the
first position of the method call.
However,
result = All.FindByParameterName (“productName”, 1);

would result in 1 item:
             PrintOrderDetails
Because there’s an argument on the second position(zero based) called
“productName”
```

## 5.65 CxList.FindByParameters Method (CxList)

Returns a CxList which is a subset of “this” instance and its elements are methods of the given CxList with the specified parameters.

### Syntax

```
CxQL
public CxList FindByParameters (CxList paramList)
```

**Parameters**

**paramList**

CxList of method parameters. GetArrayOfNodeIds

**Return Value**

A subset of “this” instance with methods whose parameters are given in the list.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByParameters() method.
 The input source code is:

 foo("myVar");

CxList var = All.FindByShortName("myVar");
result = All. FindByParameters(var);

 the result would consist of 1 item:
           foo
```

## 5.66 CxList.FindByParameterValue Method (int, string, BinaryOperator)

Returns a CxList which is a subset of “this” instance with methods where a given parameter number is equal (or not) to the specified value.

### Syntax

```
CxQL
public CxList FindByParameterValue(int ParamNo, string ParamValue,
                                   BinaryOperator opr)
```

**Parameters**

**ParamNo**

Zero-based index of the parameter

**ParamValue**

The value of the parameter

**BinaryOperator**

One of the followings values:

```
        BinaryOperator.IdentityEquality
    BinaryOperator.IdentityInequality
```

**Return Value**

Returns a CxList which is a subset of "this" instance with methods where a given parameter number is equal (or not) to the specified value.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByParameterValue() method.
 The input source code is:
 a = Method("Val1", 1);
 a = Method("val2", 2);
result = All.FindByParameterValue(0,"val1",BinaryOperator.IdentityEquality);
      the result would consist of 1 item:
             Method (in a = Method("Val1", 1))
result=All.FindByParameterValue(0,"val1",BinaryOperator.IdentityInequality);
      the result would consist of 1 item:
             Method (in a = Method("Val2", 2))
result=All.FindByParameterValue(1,"2",BinaryOperator.IdentityEquality);
      the result would consist of 1 item:
             Method (in a = Method("Val2", 2))
```

## 5.67 CxList.FindByParameterValue Method (int, string, BinaryOperator, bool)

Returns a CxList which is a subset of "this" instance with methods where a given parameter number is equal (or not) to the specified value.

### Syntax

```
CxQL
public CxList FindByParameterValue(int ParamNo, string ParamValue,
                                   BinaryOperator opr, bool useAbstractValue)
```

**Parameters**

**ParamNo**

Zero-based index of the parameter

**ParamValue**

The value of the parameter

**BinaryOperator**

One of the followings values:

```
BinaryOperator.IdentityEquality
BinaryOperator.IdentityInequality
```

**useAbstractValue**

Enable AbsInt when checking the parameter value (default : false)

**Return Value**

Returns a CxList which is a subset of "this" instance with methods where a given parameter number is equal (or not) to the specified value.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByParameterValue() method.
 The input source code is:
 a = Method("Val1", 1);
 a = Method("val2", 2);
result =
All.FindByParameterValue(0,"val1",BinaryOperator.IdentityEquality,true);
      the result would consist of 1 item:
             Method (in a = Method("Val1", 1))
result=All.FindByParameterValue(0,"val1",BinaryOperator.IdentityInequality,true)
;
      the result would consist of 1 item:
             Method (in a = Method("Val2", 2))
result=All.FindByParameterValue(1,"2",BinaryOperator.IdentityEquality,true);
      the result would consist of 1 item:
             Method (in a = Method("Val2", 2))
```

## 5.68 CxList.FindByParameterValue Method (int, int, BinaryOperator)

Returns a CxList which is a subset of "this" instance and its elements are methods whose parameters values (referred by their index) are equal (or not).

### Syntax

```
CxQL
```

```
public CxList FindByParameterValue(int paramNo1, int paramNo2, BinaryOperator
opr)
```

**Parameters**

**paramNo1**

Zero-based index of the parameter.

**paramNo2**

Zero-based index of the parameter.

**opr**

One of the following values:

```
        BinaryOperator.IdentityEquality
    BinaryOperator.IdentityInequality
```

**Return Value**

A subset of "this" instance whose parameter values are equal or not equal (depending on the operator choosen).

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.GetParameters() method.
 The input source code is:

foo(1, i, 1);

CxList methods = All.FindByType(typeof(MethodInvokeExpr));
result = All. FindByParameterValue(0, 2, BinaryOperator.IdentityEquality);

 the result would consist of 1 item:
             foo (first parameter value is equal to the third one)
```

## 5.69 CxList.FindParameterByName Method (string)

Returns a CxList which is a subset of "this" instance and its elements are parameters of the given CxList with the specified parameters.

### Syntax

```
CxQL
public CxList FindParameterByName (string paramName)
```

**Parameters**

**paramName**

String containing the name of the parameter.

**Return Value**

A subset of "this" instance where its elements are Params and these contains arguments labelled according to 'paramName'.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL
 This example demonstrates the CxList.FindParameterByName() method.
 The input source code is:

 foo(param1:”myVar”);

 result = All. FindParameterByName(“param1”);

 The result would consist of 1 item:
       param1
```

### Version Information

Supported from v9.2.0

## 5.70 CxList.FindByNumberOfParameters Method (int)

Returns a CxList which is a subset of “this” instance and its elements are nodes (method declarations, constructor, etc) with the number of parameters given.

### Syntax

```
CxQL
public CxList FindByNumberOfParameters (int numParams)
```

**Parameters**

**numParams**

int with the number of parameters.

**Return Value**

A subset of this instance with nodes (method declarations, constructor, etc) that has the number of parameters given.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL
 This example demonstrates the CxList.FindByNumberOfParameters() method.
 The input source code is:

 foo("myVar");

CxList var = All.FindByShortName("myVar");
result = All.FindByNumberOfParameters(1);
```

```
 the result would consist of 1 item:
    foo
```

### Version Information

Supported from v9.4

## 5.71 CxList.FindByPointerType Method (string, int, bool)

Returns a CxList which is a subset of "this" instance and its elements are of the type pointer of the specified type of code element

### Syntax

```
CxQL
public CxList FindByPointerType(string type, int maxDepth, bool caseSensitive)
```

**Parameters**

**type**

type of the parameter

**maxDepth**

Zero-based maximum depth to look for. Default value is 0 meaning that it will look all the PointerTypeRef levels ultil it finds a TypeRef

**caseSensitive**

Default value is true

**Return Value**

A subset of "this" instance and its elements are of the type pointer of the specified type of code element.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **type is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByPointerType method.
 The input source code is:

Var i *int

result = All.FindByPointerType("int");

 the result would consist of 2 items:
             *int - pointer
              i - Declarator
```

### Version Information

Supported from v8.5.0

## 5.72 CxList.FindByPointerType Method (string, bool)

Returns a CxList which is a subset of “this” instance and its elements are of the type pointer of the specified type of code element

### Syntax

```
CxQL
public CxList FindByPointerType(string type, bool caseSensitive)
```

**Parameters**

**type**

type of the parameter

**caseSensitive**

Default value is true

**Return Value**

A subset of “this” instance and its elements are of the type pointer of the specified type of code element.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **type is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByPointerType method.
 The input source code is:

Var i *int

result = All.FindByPointerType(“int”);

 the result would consist of 2 items:
             *int – pointer
              I - Declarator
```

### Version Information

Supported from v8.5.0

## 5.73 CxList.FindByPointerTypes Method (string[], int, bool)

Returns a CxList which is a subset of “this” instance and its elements are of the type pointer of the specified types of code element

### Syntax

```
CxQL
public CxList FindByPointerType(string[] type, int maxDepth, bool caseSensitive)
```

**Parameters**

**type**

types of the parameter

**maxDepth**

Zero-based maximum depth to look for. Default value is 0 meaning that it will look all the PointerTypeRef levels ultil it finds a TypeRef

**caseSensitive**

Default value is true

**Return Value**

A subset of "this" instance and its elements are of the type pointer of the specified types of code element.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **type is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByPointerType method.
 The input source code is:

Var i *int

result = All.FindByPointerTypes(new string[]{"int","string"});

 the result would consist of 2 items:
            *int - pointer
             i - Declarator
```

### Version Information

Supported from  v8.5.0

## 5.74 CxList.FindByPointerTypes Method (string[], bool)

Returns a CxList which is a subset of "this" instance and its elements are of the type pointer of the specified types of code element

### Syntax

```
CxQL
public CxList FindByPointerType(string[] type, bool caseSensitive)
```

**Parameters**

**type**

types of the parameter

**caseSensitive**

Default value is true

**Return Value**

A subset of “this” instance and its elements are of the type pointer of the specified types of code element.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **type is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByPointerType method.
 The input source code is:

Var i *int

result = All.FindByPointerTypes(new string[]{“int”,”string”});

 the result would consist of 2 items:
             *int - pointer
              i - Declarator
```

### Version Information

Supported from  v8.5.0

## 5.75 CxList.FindByPosition Method (int)

Returns a CxList which is a subset of “this” instance and its elements are in the given line number.

### Syntax

```
CxQL
public CxList FindByPosition(int line)
```

**Parameters**

**line**

The line number.

**Return Value**

A subset of “this” instance with elements from the given line.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL
```

```
 This example demonstrates the CxList. FindByPosition() method.
 The input source code is:

 int b, a = 5;
 if (a > 3)
      b = 6;

result = All.FindByPosition(2);

 the result would consist of 4 items:
            if
            a,
            >,
            3
```

## 5.76 CxList.FindByPosition Method (int, int)

Returns a CxList which is a subset of “this” instance and its elements are located in the given line and column number.

### Syntax

```
CxQL
public CxList FindByPosition(int line, int col)
```

**Parameters**

**Line**

Line number in the source code.

**Col**

Column number in the source code.

**Return Value**

A subset of “this” instance with elements from the given line and column.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByPosition() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();

result = All.FindByPosition (3, 16);
 the result would consist of 1 item:
             3 (in a.DataMember = 3)
```

## 5.77 CxList.FindByPosition Method (int, int, int)

Returns a CxList which is a subset of "this" instance and its elements are in the given line/column and with the given length.

### Syntax

```
CxQL
public CxList FindByPosition(int line, int col, int length)
```

**Parameters**

**line**

The line number.

**col**

The column number.

**length**

The element length.

**Return Value**

A subset of "this" instance with elements from the given line, column and with the given length.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList. FindByPosition() method.
 The input source code is:

 int b, a = 5;
 if (a == 33)
      b = 6;

result = All.FindByPosition(2, 5, 1);

 the result would consist of 1 item:
           a
```

## 5.78 CxList.FindByPosition Method (string, int)

Returns a CxList which is a subset of "this" instance and its elements are located in the given file and line number.

### Syntax

```
CxQL
public CxList FindByPosition(string file, int line)
```

**Parameters**

**file**

File name in the source code.

**line**

Line number in the source code.

**Return Value**

A subset of “this” instance which is located in the given file and line.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindByPosition() method.
 The input source code is (file name "Mycode.java"):

 MyClass a;
 int b;
 a.DataMember = 5;
 b = a.Method();

result = All.FindByPosition (“MyCode.java”, 3);
 the result would consist of 1 item:
       5 (in a.DataMember = 5)
```

## 5.79 CxList.FindByPosition Method (string, int, int)

Returns a CxList which is a subset of “this” instance and its elements are located in the given file, line and column.

### Syntax

```
CxQL
public CxList FindByPosition(string file, int line, int col)
```

**Parameters**

**file**

File name in the source code.

**line**

Line number in the source code.

**col**

Column number in the source code.

**Return Value**

A subset of “this” instance which is located in the given file, line and column.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

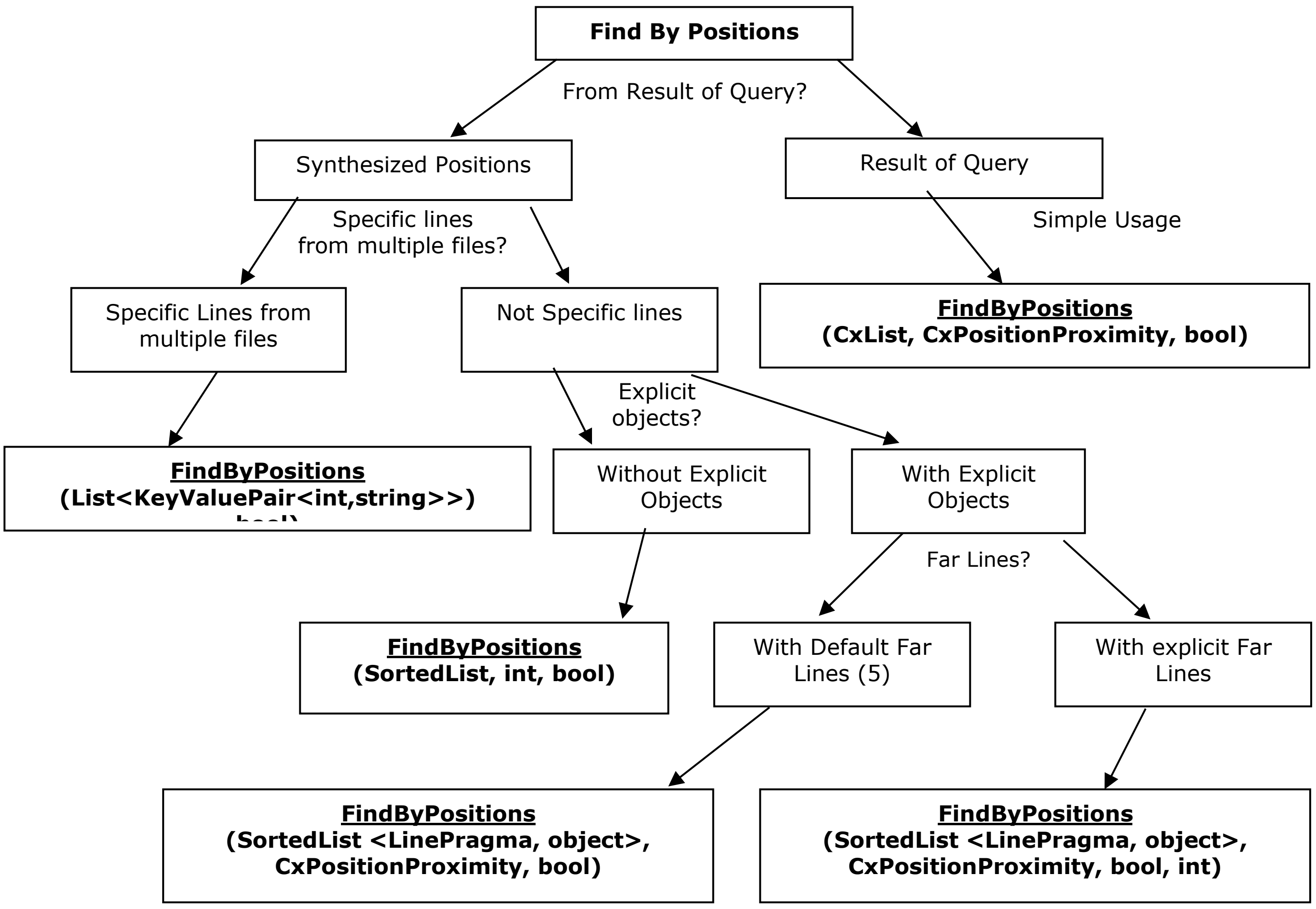
### Example

```
CxQL

 This example demonstrates the CxList.FindByPosition() method.
file name "Mycode.java"
 The input source code is:
 MyClass a;
 int b;
 a.DataMember = 5;
 b = a.Method();

result = All.FindByPosition (“MyCode.java”, 3, 16);
 the result would be -
      1 item found:
             5 (in a.DataMember = 5)
```

## 5.80 CxList.FindByPositions Methods

There are four methods (atomic queries) for using the "Find By Positions" method CxQL.

The recommended selection between the possible methods should be done according to the following tree:


Find By Positions
From Result of Query?
Synthesized Positions
Result of Query
Simple Usage
Specific lines from multiple files?
Specific Lines from multiple files
Not Specific lines
FindByPositions (CxList, CxPositionProximity, bool)
Explicit objects?
FindByPositions (List<KeyValuePair<int,string>>) bool)
Without Explicit Objects
With Explicit Objects
Far Lines?
FindByPositions (SortedList, int, bool)
With Default Far Lines (5)
With explicit Far Lines
FindByPositions (SortedList <LinePragma, object>, CxPositionProximity, bool)
FindByPositions (SortedList <LinePragma, object>, CxPositionProximity, bool, int)

### 5.80.1 CxList.FindByPositions Method (SortedList, int, bool)

Finds the elements of “this” instance at positions given in the pragmas list.

#### Syntax

```
CxQL
public CxList FindByPositions(SortedList pragmas, int extendMatch, bool oneOnly)
```

**Parameters**

**pragmas**

A sorted list containing the pragmas to match.

**extendMatch**

Defines the closeness of the matching results:

0 => ExactMatch: find exact match

1 => FindInLine: extend search to objects in closest position within same line

2 => FindClosestMatch: extend match to closest position within the same file

**oneOnly**

If true, it returns one result per position.

**Return Value**

The elements from “this” instance that are at the required positions.

#### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **First parameter is a null reference** |

#### Comments

The return value may be empty (Count = 0).

#### Example

```
CxQL

 This example demonstrates the CxList. FindByPositions() method.
 The input source code is:

 int b, a = 5;
 if (a == 33)
      b = 6;

CxList list = All.FindByShortName("b");
SortedList sorted = new SortedList(new PragmaComparer());
foreach (KeyValuePair<int, IGraph> dic in list.data)
{
       sorted.Add(dic.Value.LinePragma, null);
}
result = All.FindByPositions(sorted, 1, true);

the result would consist of 2 items:
         b (in int b)
         b (in b = 6)
```

### 5.80.2 CxList.FindByPositions Method (CxList, CxPositionProximity, bool)

Finds the elements of “this” instance at positions given in the list using the proximity given in parameter.

#### Syntax

```
CxQL
```

```
public CxList FindByPositions(CxList positions, CxPositionProximity extendMatch,
bool oneOnly)
```

**Parameters**

**positions**

A list containing the pragmas to match.

**extendMatch**

Defines the closeness of the matching results. One of the following values:

ExactMatch: find exact match

FindInLine: extend search to objects in closest position within same line

FindClosestMatch: extend match to closest position within the same file

**oneOnly**

If true, it returns one result per position.

**Return Value**

The elements of "this" instance that are at the given positions.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **First parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList. FindByPositions() method.
 The input source code is:
 int b, a = 5;
 if (a == 33)
      b = 6;

CxList list = All.FindByName("b");
result = All.FindByPositions(list, CxPositionProximity.FindInLine, false);

 the result would be all the elements in the 5 lines closer to lines that appear
variable b –
    2 items found
         b (in int b)
         b (in b = 6)
```

## 5.80.3 CxList.FindByPositions Method (SortedList<LinePragma,object>, CxPositionProximity, bool)

Finds the elements of "this" instance at positions given in the pragmas list using the proximity from the parameter.

### Syntax

```
CxQL
public CxList FindByPositions(SortedList<LinePragma,object> pragmas,
CxPositionProximity extendMatch, bool oneOnly)
```

**Parameters**

**pragmas**

A sorted list containing the pragmas to match.

**extendMatch**

Defines the closeness of the matching results. One of the following values:

FindInLine: extend search to objects in closest position within same line.

FindClosestMatch: extend match to closest position within the same file.

ExactMatch: find exact match.

**oneOnly**

If true, it returns one result per position.

**Return Value**

The elements from the current instance that are at the given positions.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList. FindByPositions() method.
 The input source code is:

 int b, a = 5;
 if (a == 33)
      b = 6;

CxList list = All.FindByName("b");
SortedList<LinePragma, object> sorted =
new SortedList<LinePragma, object>(new DataCollections.PragmaComparer());

foreach (KeyValuePair<int, IGraph> dic in list.data) {
       sorted.Add(dic.Value.LinePragma, null);
}

result = All.FindByPositions(sorted, CxList.CxPositionProximity.FindInLine,
true);

 the result would consist of 2 items:
        b (in int b)
        b (in b = 6)
```

## 5.80.4 CxList.FindByPositions Method (SortedList<LinePragma,object>, CxPositionProximity, bool, int)

Finds the elements of "this" instance at positions given in the pragmas list using the proximity given in parameter.

### Syntax

```
CxQL
```

```
public CxList FindByPositions(SortedList<LinePragma,object> pragmas,
CxPositionProximity extendMatch, bool oneOnly, int farLines)
```

**Parameters**

**pragmas**

A sorted list containing the pragmas to match.

**extendMatch**

Defines the closeness of the matching results. One of the following values:

FindInLine: extend search to objects in closest position within same line.

FindClosestMatch: extend match to closest position within the same file.

ExactMatch: find exact match.

**oneOnly**

If true, it returns one result per position.

**farLines**

Acceptable line distance to look for (the default recomended setting is 5).

**Return Value**

The elements from "this" instance that are at the given positions.

## Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

## Remarks

The return value may be empty (Count = 0).

## Example

```
This example demonstrates the CxList. FindByPositions() method.
 The input source code is:

 int b, a = 5;
 if (a == 33)
      b = 6;

CxList list = All.FindByName("b");
SortedList<LinePragma, object> sorted =
new SortedList<LinePragma, object>(new DataCollections.PragmaComparer());

foreach (KeyValuePair<int, IGraph> dic in list.data) {
       sorted.Add(dic.Value.LinePragma, null);
}

result = All.FindByPositions(sorted, CxList.CxPositionProximity.FindInLine, true,
5);

 the result would consist of 2 items:
        b (in int b)
        b (in b = 6)
```

## 5.80.5 CxList.FindByPositions Method (List<KeyValuePair<int, string>>)

Finds the elements of "this" instance at lines of files given in parameter.

### Syntax

```
CxQL
public CxList FindByPositions(List<KeyValuePair<int, string>> lines)
```

**Parameters**

**lines**

A list of pairs line/filename to search the elements.

**Return Value**

The subset of elements from "this" instance that are in the files given at the lines requested.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

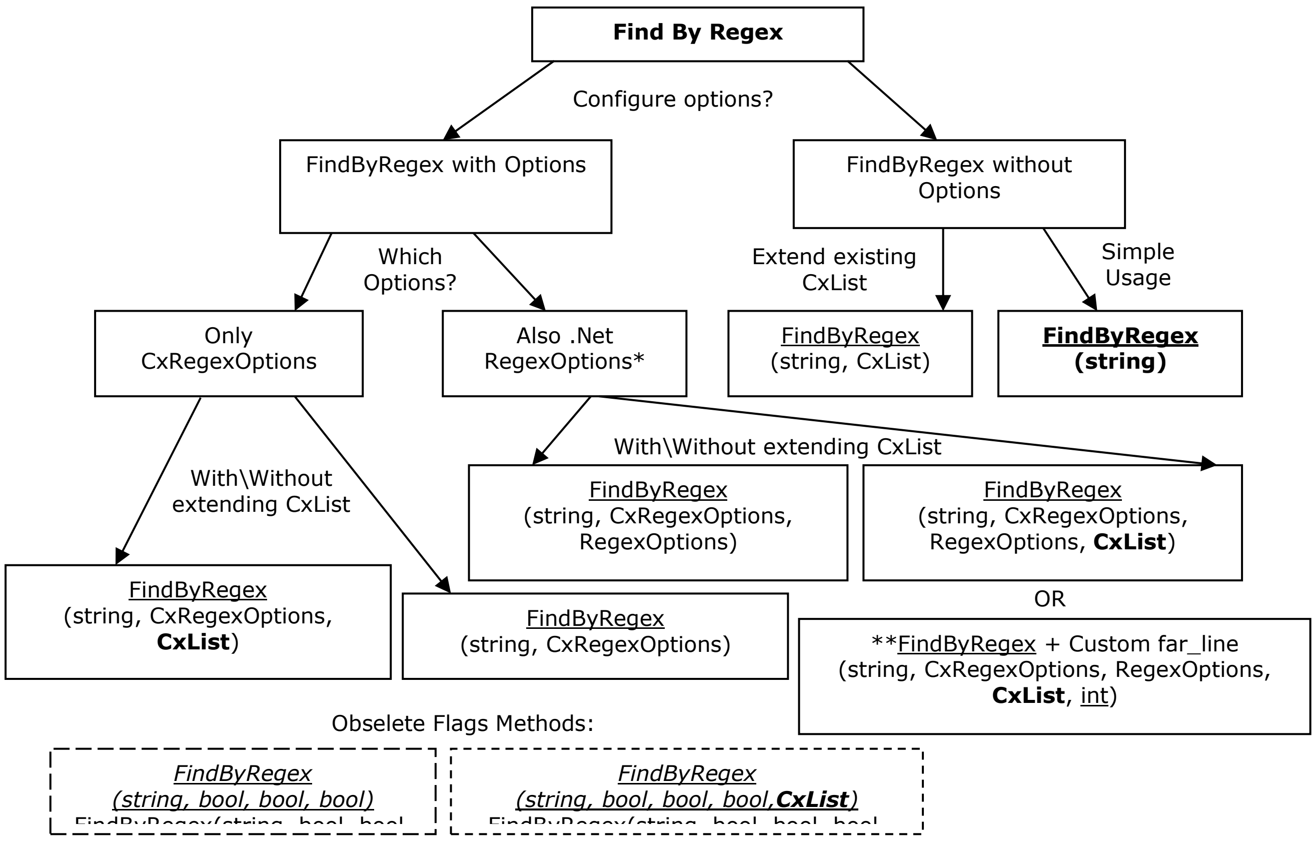
### Example

```
CxQL

 This example demonstrates the CxList. FindByPositions() method.
 The input source code is (file name is "MyCode.cs":

  int b, a = 5;
  if (a == 33)
      b = 6;

KeyValuePair<int,string> position= new
KeyValuePair<int,string>(3,"path\\MyCode.cs");
List<KeyValuePair<int,string>> list = new List<KeyValuePair<int,string>>();
list.Add(position);
result = All.FindByPositions(list);

 the result would consist of 3 items:
            b
            =
            6
```

## 5.81 CxList.FindByRegex Methods

There are few methods (atomic queries) for using the "Find By Regex" algorithm in CxQL, some of them are obsolete and not recommended, and some of them are more comfortable according to the required parameters scenario.

The recommended selection between the possible methods should be done according to the following tree:


Find By Regex
Configure options?
FindByRegex with Options
FindByRegex without Options
Which Options?
Extend existing CxList
Simple Usage
Only CxRegexOptions
Also .Net RegexOptions*
FindByRegex (string, CxList)
FindByRegex (string)
With\Without extending CxList
With\Without extending CxList
FindByRegex (string, CxRegexOptions, RegexOptions)
FindByRegex (string, CxRegexOptions, RegexOptions, CxList)
FindByRegex (string, CxRegexOptions, CxList)
FindByRegex (string, CxRegexOptions)
OR
**FindByRegex + Custom far_line (string, CxRegexOptions, RegexOptions, CxList, int)
Obselete Flags Methods:
FindByRegex (string, bool, bool, bool)
FindByRegex (string, bool, bool, bool,CxList)

- Even without mentioning it explicitly in the parameter, the RegexOptions.**Multiline**, and RegexOptions.**Singleline** are always enabled in the Find-By-Regex algorithm in these queries.
- Customizing the FAR_LINES parameter is possible using the new method (the default value of this parameter is 5 and it is relevant for searching regex matches in comments).
- The full path (including namespaces) of the CxRegexOptions enum is CxList.CxRegexOptions.
- The full path (including namespaces) of the RegexOptions enum is System.Text.RegularExpressions.RegexOptions.

## 5.81.1 CxList.FindByRegex Method (string)

Returns a CxList which is a subset of this instance and its elements match the specified regular expression string.

This call is equivalent to the following calls and it is recommended to use the short call format by default:

- `FindByRegex(expression, null)`
- `FindByRegex(expression, CxRegexOptions.None)`
- `FindByRegex(expression, CxRegexOptions.None, RegexOptions.None)`
- `FindByRegex(expression, CxRegexOptions.None, RegexOptions.None, null)`
- `FindByRegex(expression, CxRegexOptions.None, RegexOptions.None, null, 5)`
- `FindByRegex(expression, false, true, false)`
- `FindByRegex(expression, false, true, false, null)`
- `FindByRegex(expression, CxRegexOptions.None, null)`

### Syntax

```
CxQL
public CxList FindByRegex(string expression)
```

**Parameters**

**expression**

Regular expression string.

**Return Value**

A subset of this instance matches the given regular expression.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByRegex method.

```
CxQL

 This example demonstrates the CxList.FindByRegex() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);

result = All.FindByRegex(@"(\s)?foo\(");

 the result would be -
      1 item found:
             foo
```

### Version Information

Supported from: **CxAudit** v1.8.1

## 5.81.2 CxList.FindByRegex Method (string, bool, bool, bool)

Returns a CxList which is a subset of this instance and its elements match the specified regular expression string, according to specified flag parameters.

This call is equivalent to the following calls and it is highly recommended to use the enum instead of the confusing flags:

- `FindByRegex(expression, searchInComments, searchInStringLiterals, recursive, null)`
- The 3 flags are translated to `CxRegexOptions` enum in the following way (bitmask supported):
  - `(false, false, false) => CxRegexOptions.DoNotSearchInStringLiterals`
  - `(false, false, true) => CxRegexOptions.DoNotSearchInStringLiterals | CxRegexOptions.AllowOverlaps`
  - `(false, true, false) => CxRegexOptions.None`
  - `(false, true, true) => CxRegexOptions.AllowOverlaps`
  - `(true, false, false) => CxRegexOptions.SearchInComments |`

```
                CxRegexOptions.DoNotSearchInStringLiterals
```
- `(true, false, true) =>`
  ```
        CxRegexOptions.SearchInComments |
        CxRegexOptions.DoNotSearchInStringLiterals |
        CxRegexOptions.AllowOverlaps
  ```
- `(true, true, false) => CxRegexOptions.SearchInComments`
- `(true, true, true) =>`
  ```
        CxRegexOptions.SearchInComments |
        CxRegexOptions.AllowOverlaps
  ```

After translating the flags to CxRegexOptions enum this call is equivalent to the following calls:

- `FindByRegex(expression, cxRegexOptions)`
- `FindByRegex(expression, cxRegexOptions, RegexOptions.None)`
- `FindByRegex(expression, cxRegexOptions, RegexOptions.None, null)`
- `FindByRegex(expression, cxRegexOptions, RegexOptions.None, null, 5)`
- `FindByRegex(expression, cxRegexOptions, null)`

## Syntax

```
CxQL
public CxList FindByRegex(string expression , bool searchInComments, bool
searchInStringLiterals, bool recursive)
```

**Parameters**

**expression**

Regular expression string.

**searchInComments**

Positive if searching inside comments is desired.

**searchInStringLiterals**

Positive if searching inside string literals is desired.

**recursive**

Positive if it is desired to allow regex matches to overlap.

**Return Value**

A subset of this instance matches the given regular expression according to the additional parameters.

## Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **Expression parameter is a null reference** |

## Remarks

The return value may be empty (Count = 0).

## Example

The following code example shows how you can use the FindByRegex method.

```
CxQL

 This example demonstrates the CxList.FindByRegex() method.
 The input source code is:
```

```
 int a = 5;
 if (a > 3)
     foo(a);

result = All.FindByRegex(@"(\s)?foo\(", false, true, false);

 the result would be -
      1 item found:
             foo
```

### Version Information

Supported from v1.8.1

## 5.81.3 CxList.FindByRegex Method (string, bool, bool, bool, CxList)

Returns a CxList which is a subset of this instance and its elements match the specified regular expression string, according to specified flag parameters and fill the extended results parameter with the strings of the matches.

- The 3 flags are translated to `CxRegexOptions` enum in the following way (bitmask supported):
  - `(false, false, false) => CxRegexOptions.DoNotSearchInStringLiterals`
  - `(false, false, true) =>`
    `CxRegexOptions.DoNotSearchInStringLiterals |`
    `CxRegexOptions.AllowOverlaps`
  - `(false, true, false) => CxRegexOptions.None`
  - `(false, true, true) => CxRegexOptions.AllowOverlaps`
  - `(true, false, false) =>`
    `CxRegexOptions.SearchInComments |`
    `CxRegexOptions.DoNotSearchInStringLiterals`
  - `(true, false, true) =>`
    `CxRegexOptions.SearchInComments |`
    `CxRegexOptions.DoNotSearchInStringLiterals |`
    `CxRegexOptions.AllowOverlaps`
  - `(true, true, false) => CxRegexOptions.SearchInComments`
  - `(true, true, true) =>`
    `CxRegexOptions.SearchInComments | CxRegexOptions.AllowOverlaps`

After translating the flags to `CxRegexOptions` enum this call is equivalent to the following calls:

**(It is highly recommended to use the enum instead of the confusing flags)**

- `FindByRegex(expression, cxRegexOptions, RegexOptions.None, cxList)`
- `FindByRegex(expression, cxRegexOptions, RegexOptions.None, cxList, 5)`
- `FindByRegex(expression, cxRegexOptions, cxList)`

### Syntax

```
CxQL
public CxList FindByRegex(string expression , bool searchInComments, bool
searchInStringLiterals, bool recursive, CxList extendedResults)
```

**Parameters**

**expression**

Regular expression string.

**searchInComments**

Positive if searching inside comments is desired.

**searchInStringLiterals**

Positive if searching inside string literals is desired.

**recursive**

Positive if it is desired to allow regex matches to overlap.

**extendedResults**

extendedResults parameter is filled with the strings of the matches.

**Return Value**

A subset of this instance matches the given regular expression according to the additional parameters.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **Expression parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByRegex method.

```
CxQL

 This example demonstrates the CxList.FindByRegex() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);

result = All.FindByRegex(@"(\s)?foo\(", false, true, false, All.NewCxList());

 the result would be -
      1 item found:
             foo
```

### Version Information

Supported from v1.8.1

## 5.81.4 CxList.FindByRegex Method (string, CxList)

Returns a CxList which is a subset of this instance and its elements match the specified regular expression string, and fill the extended results parameter with the strings of the matches.

This query search source files with regex, and return the closest same line DOM object to the matches.

If no such object exists, returns the closest object in a successive line.

Search does not include searching inside comments and string literals, and regex matches are not allowed to overlap. The matching strings are returned in the extendedResults paramater.

This call is equivalent to the following calls:

- `FindByRegex(expression, CxRegexOptions.None, RegexOptions.None, cxList)`

- `FindByRegex(expression, CxRegexOptions.None, RegexOptions.None, cxList, 5)`
- `FindByRegex(expression, false, true, false, cxList)`
  - Using the Boolean flags option is not recommended, use the enums instead.
- `FindByRegex(expression, CxRegexOptions.None, cxList)`

### Syntax

```
CxQL
public CxList FindByRegex(string expression , CxList extendedResults)
```

**Parameters**

**expression**

Regular expression string.

**extendedResults**

extendedResults parameter is filled with the strings of the matches.

**Return Value**

A subset of this instance matches the given regular expression according to the additional parameters.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **Expression parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByRegex method.

```
CxQL

 This example demonstrates the CxList.FindByRegex() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);

result = All.FindByRegex(@"(\s)?foo\(", All.NewCxList());

 the result would be -
      1 item found:
             foo
```

### Version Information

Supported from v1.8.1

## 5.81.5 CxList.FindByRegex Method (string, CxRegexOptions)

Returns a CxList which is a subset of this instance and its elements match the specified regular expression string, according to specified Checkmarx Regex Options defined in the second parameter.

This call is equivalent to the following calls and it is recommended to use the short call format by default:

- `FindByRegex(expression, cxRegexOptions, RegexOptions.None)`
- `FindByRegex(expression, cxRegexOptions, RegexOptions.None, null)`

- `FindByRegex(expression, cxRegexOptions, RegexOptions.None, null, 5)`
- `FindByRegex(expression, cxRegexOptions, null)`

## Syntax

```
CxQL
public CxList FindByRegex(string expression , CxRegexOptions cxOptions)
```

**Parameters**

**expression**

Regular expression string.

**cxOptions**

An enum matching the relevant CxRegexOptions which are:

None, SearchInComments, DoNotSearchInStringLiterals, AllowOverlaps and SearchOnlyInComments

**Return Value**

A subset of this instance matches the given regular expression according to the additional parameters.

## Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **Expression parameter is a null reference** |

## Remarks

The return value may be empty (Count = 0).

## Example

The following code example shows how you can use the FindByRegex method.

```
CxQL

 This example demonstrates the CxList.FindByRegex() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);

result = All.FindByRegex(@"(\s)?foo\(", CxList.CxRegexOptions.None);

 the result would be -
      1 item found:
             foo
```

## Version Information

Supported from v1.8.1

## 5.81.6 CxList.FindByRegex Method (string, CxRegexOptions, CxList)

Returns a CxList which is a subset of this instance and its elements match the specified regular expression string, according to specified Checkmarx Regex Options defined in the second parameter, and also fill the extended results parameter with the strings of the matches.

This call is equivalent to the following calls and it is recommended to use the short call format by default:

- `FindByRegex(expression, cxRegexOptions, RegexOptions.None, cxList)`
- `FindByRegex(expression, cxRegexOptions, RegexOptions.None, cxList, 5)`

## Syntax

```
CxQL
public CxList FindByRegex(string expression , CxRegexOptions cxOptions, CxList
extendedResults)
```

**Parameters**

**expression**

Regular expression string.

**cxOptions**

An enum matching the relevant CxRegexOptions which are:

None, SearchInComments, DoNotSearchInStringLiterals, AllowOverlaps and SearchOnlyInComments

**extendedResults**

extendedResults parameter is filled with the strings of the matches.

**Return Value**

A subset of this instance matches the given regular expression according to the additional parameters.

## Exceptions

| **Exception type** | **Condition** |
|---|---|
| ArgumentNullException | **Expression parameter is a null reference** |

## Remarks

The return value may be empty (Count = 0).

## Example

The following code example shows how you can use the FindByRegex method.

```
CxQL

 This example demonstrates the CxList.FindByRegex() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);

result = All.FindByRegex(@"(\s)?foo\(", CxList.CxRegexOptions.None,
All.NewCxList());

 the result would be -
      1 item found:
             foo
```

## Version Information

Supported from v1.8.1

## 5.81.7 CxList.FindByRegex Method (string, CxRegexOptions, RegexOptions)

Returns a CxList which is a subset of this instance and its elements match the specified regular expression string, according to specified Regex Options defined in the parameters (Checkmarx regex options and standard regex options).

This call is equivalent to the following calls and it is recommended to use the short call format by default:

- `FindByRegex(expression, cxRegexOptions, regexOptions, null)`
- `FindByRegex(expression, cxRegexOptions, regexOptions, null, 5)`

### Syntax

```
CxQL
public CxList FindByRegex(string expression , CxRegexOptions cxOptions,
RegexOptions regularOptions)
```

**Parameters**

**expression**

Regular expression string.

**cxOptions**

An enum matching the relevant CxRegexOptions which are:

None, SearchInComments, DoNotSearchInStringLiterals, AllowOverlaps and SearchOnlyInComments

**regularOptions**

Options to add to the regular expression (case sensitivity, etc.)

In addition to the user-defined regular-expression-options in this arguments, the alogrith also uses the following regex-options by default: `RegexOptions.Multiline, RegexOptions.Singleline.`

**Return Value**

A subset of this instance matches the given regular expression according to the additional parameters.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **Expression parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByRegex method.

```
CxQL

 This example demonstrates the CxList.FindByRegex() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);

result = All.FindByRegex(@"(\s)?foo\(", CxList.CxRegexOptions.None,
System.Text.RegularExpressions.RegexOptions.None);

 the result would be -
```

```
      1 item found:
             foo
```

### Version Information

Supported from v1.8.1

## 5.81.8 CxList.FindByRegex Method (string, CxRegexOptions, RegexOptions, CxList)

Returns a CxList which is a subset of this instance and its elements match the specified regular expression string, according to specified Regex Options defined in the parameters (Checkmarx regex options and standard regex options), and also fill the extended results parameter with the strings of the matches.

This call is equivalent to the following call and it is recommended to use the short call format by default:

- `FindByRegex(expression, cxRegexOptions, regexOptions, cxList, 5)`

### Syntax

```
CxQL
public CxList FindByRegex(string expression , CxRegexOptions cxOptions,
RegexOptions regularOptions, CxList extendedResults)
```

**Parameters**

**expression**

Regular expression string.

**cxOptions**

An enum matching the relevant CxRegexOptions which are:

None, SearchInComments, DoNotSearchInStringLiterals, AllowOverlaps and SearchOnlyInComments

**regularOptions**

Options to add to the regular expression (case sensitivity, etc.)

In addition to the user-defined regular-expression-options in this arguments, the alogrith also uses the following regex-options by default: `RegexOptions`.Multiline, `RegexOptions`.Singleline.

**extendedResults**

extendedResults parameter is filled with the strings of the matches.

**Return Value**

A subset of this instance matches the given regular expression according to the additional parameters.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **Expression parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByRegex method.

```
CxQL

 This example demonstrates the CxList.FindByRegex() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);

result = All.FindByRegex(@"(\s)?foo\(", CxList.CxRegexOptions.None,
System.Text.RegularExpressions.RegexOptions.None, All.NewCxList());

 the result would be -
      1 item found:
             foo
```

### Version Information

Supported from v1.8.1

## 5.81.9 CxList.FindByRegex Method (string, CxRegexOptions, RegexOptions, CxList, int, CxPositionSearchDirection)

Returns a CxList which is a subset of this instance and its elements match the specified regular expression string, according to specified Regex Options defined in the parameters (Checkmarx regex options and standard regex options), and also fill the extended results parameter with the strings of the matches.

Also get a customized far-lines parameter to be considered as acceptable lines distance when looking for regex in comments.

All the other calls to "FindByRegex.." with\without different parameters lead in the end to this specific method.

### Syntax

CxQL
```
public CxList FindByRegex(string expression , CxRegexOptions cxOptions,
RegexOptions regularOptions, CxList extendedResults, int farLines,
CxPositionSearchDirection searchDirection )
```

**Parameters**

**expression**

Regular expression string.

**cxOptions**

An enum matching the relevant CxRegexOptions which are:

None, SearchInComments, DoNotSearchInStringLiterals, AllowOverlaps and SearchOnlyInComments

**regularOptions**

Options to add to the regular expression (case sensitivity, etc.)

In addition to the user-defined regular-expression-options in this arguments, the alogrith also uses the following regex-options by default: `RegexOptions`.Multiline, `RegexOptions`.Singleline.

**extendedResults**

extendedResults parameter is filled with the strings of the matches.

**farLines**

Configure the line distance to look for regex matches in comments (it is 5 lines by default).

**searchDirection**

Determines the search direction that can be one of the following values: Default, Backward, Forward. The Backward and Forward values means that the search is for the CxList which is a subset of the instance that is the last one before the regular expression or just the first one after, respectively. The default search (that is the default value of the parameter) just compares the distance between both (in manner of Line distance and column distance) and chooses the one that is the closest between the two.

**Return Value**

A subset of this instance matches the given regular expression according to the additional parameters.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **Expression parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByRegex method.

```
CxQL

 This example demonstrates the CxList.FindByRegex() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);

result = All.FindByRegex(@"(\s)?foo\(", CxList.CxRegexOptions.None,
System.Text.RegularExpressions.RegexOptions.None, All.NewCxList(), 5);

 the result would be -
      1 item found:
             foo
```

### Version Information

Supported from v1.8.1

## 5.81.10 CxList.FindByRegexSecondOrder Method (string, CxList)

Filters a CxList of Comments DOM objects according to a check of whether a Comment object contain a match to the provided regex expression, and returns closest DOM object to those that pass the filter.

Used in C\C++ MISRA Preset queries in order to validate comments style.

### Syntax

```
CxQL
public CxList FindByRegexSecondOrder(string expression , CxList extendedResults)
```

**Parameters**

**expression**

Regular expression search string.

**inputList**

The comments CxList that's should be filtered.

**Return Value**

A subset of this instance matches the given regular expression according to the additional parameters.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **Expression parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByRegexSecondOrder method

```
CxQL

 This example demonstrates the CxList.FindByRegexSecondOrder() method.
 The input source code is taken from MISRA Code_Commented_Out query:

/* Function comment is compliant. * /
void mc2_0202 ( void )
{
use_int32(0);   // Comment Not Compliant
}
*/

// Find all comments ending with } or ;
CxList extendedResult = All.NewCxList();

// All /* */ comments
CxList res = All.FindByRegex(@"/\*.*?\*/", true, false, false, extendedResult);

// Search results for } or ; at end of comment
result = All.FindByRegexSecondOrder(@"[;{}]\s*\*/", extendedResult);
The result will be the commented out function which is found out by this regex
```

### Version Information

Supported from v1.8.1

## 5.82 CxList.FindByRegexExt Methods

Find by regular expression in all files of the project regardless of DOM and language.

### Remarks

The results are not related to DOM so they can't be compared to DOM objects returned by other functions. Results can't be used as parameters to other queries.

### 5.82.1 CxList.FindByRegexExt Method (string)

### Syntax

```
CxQL
public CxList FindByRegexExt(string pattern)
```

**Parameters**

**Pattern**

Regular expression pattern

**Return Value**

A list of matches for given regular expression in all project files.

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);

// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo");

 The result would be -
      3 items found:
             foo
             // foo
             /* foo
```

### Version Information

Supported from version 7.1.8 and 7.1.6HF5

## 5.82.2 CxList.FindByRegexExt Method (string, string)

### Syntax

```
CxQL
public CxList FindByRegexExt(string pattern, string fileMask)
```

**Parameters**

**Pattern**

Regular expression pattern

**fileMask**

File mask for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**Return Value**

A list of matches for given regular expression in all project files.

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:
```

```
 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);

// foo(a)
/* foo */

result = All.FindByRegexExt @"(\s)?foo", “*.cs”);

 The search would be only on *.cs files and the result would be -
      3 items found:
             foo
             // foo
             /* foo
```

### Version Information

Supported from version 7.1.8 and 7.1.6HF5

## 5.82.3 CxList.FindByRegexExt Method (string, string, string)

### Syntax

```
CxQL
public CxList FindByRegexExt(string pattern, string fileMask, string
fileExclMask)
```

**Parameters**

**Pattern**

Regular expression pattern

**fileMask**

File mask for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**fileExclMask**

File mask for excluding files with the specified extensions from the search. Control characters "*" and "?" are supported.

For example: If "*.*"is used as the value for fileMask, "*.json" can be used for fileExclMask to exclude all the JSON files from the search.

**Return Value**

A list of matches for given regular expression in all project files.

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);

// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo", ”*.*”, “*.json”);


 All files in the source code would be searched except the *.json files and the
result would be -
      3 items found:
             foo
             // foo
             /* foo
```

## Version Information

Supported from version 9.4.5.

## 5.82.4 CxList.FindByRegexExt Method (string, string, bool)

### Syntax

```
CxQL
public CxList FindByRegexExt(string pattern, string fileMask, bool
searchInComments)
```

**Parameters**

**expression**

Regular expression pattern

**fileMask**

File mask for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**searchInComments**

Allow or not search in comments

**Return Value**

A list of matches for given regular expression in chosen project files including or excluding results in comments.

### Remarks

The return value may be empty (Count = 0).

Default values relevant only from version 7.1.8.

### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);
```

```
// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo","*.cs",false);

 The result would be -
     3 items found:
             foo
             // foo
             /* foo
```

### Version Information

Supported from version 7.1.8 and 7.1.6HF5

## 5.82.5 CxList.FindByRegexExt Method (string, string, string, bool)

### Syntax

```
CxQL
public CxList FindByRegexExt(string pattern, string fileMask, string
fileExclMask, bool searchInComments)
```

**Parameters**

**expression**

Regular expression pattern

**fileMask**

File mask for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**fileExclMask**

File mask for excluding files with the specified extensions from the search. Control characters "*" and "?" are supported.

**searchInComments**

Allow or not search in comments

**Return Value**

A list of matches for given regular expression in chosen project files including or excluding results in comments.

### Remarks

The return value may be empty (Count = 0).

Default values relevant only from version 9.4.5.

### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);
 else
```

```
     FOO(a);

// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo","*.*","*.json",false);

 All files in the source code would be searched except the *.json files and the
result would be --
      1 item found:
             foo
```

### Version Information

Supported from version 9.4.5.

## 5.82.6 CxList.FindByRegexExt Method (string, string, bool, RegexOptions)

### Syntax

```
CxQL
public CxList FindByRegexExt(string pattern, string fileMask = "*.*", bool
searchInComments = true, RegexOptions regularOptions = RegexOptions.None)
```

**Parameters**

**expression**

Regular expression pattern

**fileMask**

Default value: "*.*".

File mask for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**searchInComments**

Default value: true.

Allow or not search in comments

**regularOptions**

Default value: RegexOptions.None.

Options for regular expression build from first parameter

**Return Value**

A list of matches for given regular expression in chosen project files including or excluding results in comments with regex build with specified options.

### Remarks

The return value may be empty (Count = 0).

Default values relevant only from version 7.1.8.

### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:
```

```
 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);

// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo","*.cs",false, RegexOptions.IgnoreCase);

 The result would be -
      2 items found:
             foo
             FOO
```

### Version Information

Supported from version 7.1.8 and 7.1.6HF5

## 5.82.7 CxList.FindByRegexExt Method (string, string, string, bool, RegexOptions)

### Syntax

CxQL
```
public CxList FindByRegexExt(string pattern, string fileMask = "*.*", string
fileExclMask = "", bool searchInComments = true, RegexOptions regularOptions =
RegexOptions.None)
```

**Parameters**

**expression**

Regular expression pattern

**fileMask**

Default value: "*.*".

File mask for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**fileExclMask**

File mask for excluding files with the specified extensions from the search. Control characters "*" and "?" are supported.

**searchInComments**

Default value: true.

Allow or not search in comments

**regularOptions**

Default value: RegexOptions.None.

Options for regular expression build from first parameter

**Return Value**

A list of matches for given regular expression in chosen project files including or excluding results in comments with regex build with specified options.

### Remarks

The return value may be empty (Count = 0).

Default values relevant only from version 9.4.5.

### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);

// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo","*.*","*.json",false,
RegexOptions.IgnoreCase);

 All files in the source code would be searched except the *.json files and the
result would be -
      2 items found:
             foo
             FOO
```

### Version Information

Supported from version 9.4.5.

## 5.82.8 CxList.FindByRegexExt Method (string, string, bool, CxRegexOptions, RegexOptions)

### Syntax

```
CxQL
public CxList FindByRegexExt(string pattern, string fileMask = "*.*", bool
searchInComments = true, CxRegexOptions cxOptions = CxRegexOptions.None,
RegexOptions regularOptions = RegexOptions.None)
```

**Parameters**

**expression**

Regular expression pattern

**fileMask**

Default value: "*.*".

File mask for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**searchInComments**

Default value: true.

Allow or not search in comments

**cxOptions**

An enum matching the relevant CxRegexOptions which are:

None, SearchInComments, DoNotSearchInStringLiterals, AllowOverlaps and SearchOnlyInComments

**regularOptions**

Default value: RegexOptions.None.

Options for regular expression build from first parameter.

**Return Value**

A list of matches for given regular expression in chosen project files including or excluding results in comments with regex build with specified options.

### Remarks

The return value may be empty (Count = 0).

Default values relevant only from version 7.1.8.

### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);

// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo","*.*",false, CxRegexOptions.None,
RegexOptions.IgnoreCase);

 All files in the source code would be searched and the result would be -
      2 items found:
             foo
             FOO
```

### Version Information

Supported from version 7.1.8 and 7.1.6HF5

## 5.82.9 CxList.FindByRegexExt Method (string, string, string, bool, CxRegexOptions, RegexOptions)

### Syntax

```
CxQL
public CxList FindByRegexExt(string pattern, string fileMask = "*.*", string
fileExclMask = "", bool searchInComments = true, CxRegexOptions cxOptions =
CxRegexOptions.None, RegexOptions regularOptions = RegexOptions.None)
```

**Parameters**

**expression**

Regular expression pattern

**fileMask**

Default value: "*.*".

File mask for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**fileExclMask**

File mask for excluding files with the specified extensions from the search. Control characters "*" and "?" are supported.

**searchInComments**

Default value: true.

Allow or not search in comments

**cxOptions**

An enum matching the relevant CxRegexOptions which are:

None, SearchInComments, DoNotSearchInStringLiterals, AllowOverlaps and SearchOnlyInComments

**regularOptions**

Default value: RegexOptions.None.

Options for regular expression build from first parameter.

**Return Value**

A list of matches for given regular expression in chosen project files including or excluding results in comments with regex build with specified options.

## Remarks

The return value may be empty (Count = 0).

Default values relevant only from version 9.4.5.

## Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);

// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo","*.*",”*.json”,false,
CxRegexOptions.None, RegexOptions.IgnoreCase);

 All files in the source code would be searched except the *.json files and the
result would be -
      2 items found:
             foo
             FOO
```

## Version Information

Supported from version 9.4.5.

### 5.82.10 CxList.FindByRegexExt Method (string, string, CxRegexOptions)

#### Syntax

```
CxQL
public CxList FindByRegexExt(string pattern, string fileMask, CxRegexOptions
cxOptions)
```

**Parameters**

**pattern**

Regular expression pattern

**fileMask**

File mask for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**cxOptions**

An enum matching the relevant CxRegexOptions which are:

None, SearchInComments, DoNotSearchInStringLiterals, AllowOverlaps and

SearchOnlyInComments

**Return Value**

A list of matches for given regular expression in chosen project files including or excluding results in comments with regex build with specified options.

#### Remarks

The return value may be empty (Count = 0).

Default values relevant only from version 7.1.8.

#### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);

// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo","*.*", RegexOptions.IgnoreCase);

 All files in the source code would be searched and the result would be -
      2 items found:
             foo
             FOO
```

#### Version Information

Supported from version 7.1.8 and 7.1.6HF5

### 5.82.11 CxList.FindByRegexExt Method (string, string, string, CxRegexOptions)

#### Syntax

```
CxQL
public CxList FindByRegexExt(string pattern, string fileMask, string
fileExclMask, CxRegexOptions cxOptions)
```

**Parameters**

**pattern**

Regular expression pattern

**fileMask**

File mask for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**fileExclMask**

File mask for excluding files with the specified extensions from the search. Control characters "*" and "?" are supported.

**cxOptions**

An enum matching the relevant CxRegexOptions which are:

None, SearchInComments, DoNotSearchInStringLiterals, AllowOverlaps and SearchOnlyInComments

**Return Value**

A list of matches for given regular expression in chosen project files including or excluding results in comments with regex build with specified options.

#### Remarks

The return value may be empty (Count = 0).

Default values relevant only from version 9.4.5.

#### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);

// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo","*.*",”*.json”,
RegexOptions.IgnoreCase);

 All files in the source code would be searched except the *.json files and the
result would be -
      2 items found:
             foo
             FOO
```

### Version Information

Supported from version 9.4.5.

## 5.82.12 CxList.FindByRegexExt Method (string, string, CxRegexOptions, RegexOptions)

### Syntax

```
CxQL
public CxList FindByRegexExt(string pattern, string fileMask = "*.*",
CxRegexOptions cxOptions = CxRegexOptions.None, RegexOptions regularOptions =
RegexOptions.None)
```

**Parameters**

**expression**

Regular expression pattern

**fileMask**

Default value: "*.*".

File mask for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**cxOptions**

An enum matching the relevant CxRegexOptions which are:

None, SearchInComments, DoNotSearchInStringLiterals, AllowOverlaps and SearchOnlyInComments

**regularOptions**

Default value: RegexOptions.None.

Options for regular expression build from first parameter

**Return Value**

A list of matches for given regular expression in chosen project files including or excluding results in comments with regex build with specified options.

### Remarks

The return value may be empty (Count = 0).

Default values relevant only from version 7.1.8.

### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:
 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);

// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo","*.*",false, RegexOptions.IgnoreCase);

 All files in the source code would be searched and the result would be –
```

```
      2 item found:
             foo
             FOO
```

### Version Information

Supported from version 7.1.8 and 7.1.6HF5

## 5.82.13 CxList.FindByRegexExt Method (string, string, string, CxRegexOptions, RegexOptions)

### Syntax

```
CxQL
public CxList FindByRegexExt(string pattern, string fileMask = "*.*", string
fileExclMask = "", CxRegexOptions cxOptions = CxRegexOptions.None, RegexOptions
regularOptions = RegexOptions.None)
```

**Parameters**

**expression**

Regular expression pattern

**fileMask**

Default value: "*.*".

File mask for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**fileExclMask**

File mask for excluding files with the specified extensions from the search. Control characters "*" and "?" are supported.

**cxOptions**

An enum matching the relevant CxRegexOptions which are:

None, SearchInComments, DoNotSearchInStringLiterals, AllowOverlaps and SearchOnlyInComments

**regularOptions**

Default value: RegexOptions.None.

Options for regular expression build from first parameter

**Return Value**

A list of matches for given regular expression in chosen project files including or excluding results in comments with regex build with specified options.

### Remarks

The return value may be empty (Count = 0).

Default values relevant only from version 9.4.5.

### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:
 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);
```

```
// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo","*.*","*.json",false,
RegexOptions.IgnoreCase);

 All files in the source code would be searched except the *.json files and the
result would be -
      2 item found:
             foo
             FOO
```

### Version Information

Supported from version 9.4.5.

## 5.82.14 CxList.FindByRegexExt Method (string, List<string>, bool, CxRegexOptions, RegexOptions)

### Syntax

```
CxQL
public CxList FindByRegexExt(string expression, List<string> fileMaskList, bool
searchInComments = true, CxRegexOptions cxOptions =
CxRegexOptions.SearchInComments, RegexOptions regularOptions = RegexOptions.None)
```

**Parameters**

**expression**

Regular expression pattern

**fileMaskList**

List of File masks for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**searchInComments**

Default value: true.

Allow or not search in comments

**cxOptions**

An enum matching the relevant CxRegexOptions which are:

None, SearchInComments, DoNotSearchInStringLiterals, AllowOverlaps and

SearchOnlyInComments

**regularOptions**

Default value: RegexOptions.None.

Options for regular expression build from first parameter

**Return Value**

A list of matches for given regular expression in choosen project files including or excluding results in comments with regex build with specified options.

### Remarks

The return value may be empty (Count = 0).

Default values relevant only from version 7.1.8.

### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL
```

```
 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);
// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo",new List<string>{"*.cs", "*.js"}, false,
CxList.CxRegexOptions.SearchInComments, RegexOptions.IgnoreCase);

 Files in the source code with .cs and .js extensions would be included in the
search and the result would be -
      2 items found:
             foo
             FOO
```

### Version Information

Supported from version 8.0.0

## 5.82.15 CxList.FindByRegexExt Method (string, List<string>, List<string>, bool, CxRegexOptions, RegexOptions)

### Syntax

CxQL

```
public CxList FindByRegexExt(string expression, List<string> fileMaskList,
List<string> fileExclMaskList, bool searchInComments = true, CxRegexOptions
cxOptions = CxRegexOptions.SearchInComments, RegexOptions regularOptions =
RegexOptions.None)
```

**Parameters**

**expression**

Regular expression pattern

**fileMaskList**

List of File masks for search. Control characters "*" and "?" are supported.

For example: "*.*" looks in all files and "*.aspx" looks in aspx files.

**fileExclMask**

List of File masks for excluding files with the specified extensions from the search. Control characters "*" and "?" are supported.

**searchInComments**

Default value: true.

Allow or not search in comments

**cxOptions**

An enum matching the relevant CxRegexOptions which are:

None, SearchInComments, DoNotSearchInStringLiterals, AllowOverlaps and SearchOnlyInComments

**regularOptions**

Default value: RegexOptions.None.

Options for regular expression build from first parameter

**Return Value**

A list of matches for given regular expression in choosen project files including or excluding results in comments with regex build with specified options.

### Remarks

The return value may be empty (Count = 0).

Default values relevant only from version 9.4.5.

### Example

The following code example shows how you can use the FindByRegexExt method.

```
CxQL

 This example demonstrates the CxList.FindByRegexExt() method.
 The input source code is:

 int a = 5;
 if (a > 3)
     foo(a);
 else
     FOO(a);
// foo(a)
/* foo */

result = All.FindByRegexExt(@"(\s)?foo",new List<string>{"*.cs", "*.js*"},
List<string>{"*.json"}, false, CxList.CxRegexOptions.SearchInComments,
RegexOptions.IgnoreCase);

 Files in the source code with .cs and .js* extensions would be included in the
search, except the *.json files and the result would be -
      2 items found:
             foo
             FOO
```

### Version Information

Supported from version 9.4.5.

## 5.83 CxList.FindByReturnType Method (string)

Returns a CxList which is a subset of this instance and its elements are of the specified type.

### Syntax

```
CxQL
public CxList FindByReturnType(String Type, bool stripPointerAndRefFromReturnType
= true)
```

**Parameters**

**Type**

The type of the objects to be found

**stripPointerAndRefFromReturnType**

true – the result will include methods that return Type* as well as methods that return Type

**Return Value**

A subset of this instance and its elements are of the specified return type.

### Exceptions

| Exception type | Condition |
|---|---|

| | |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByReturnType method.

```
CxQL

 This example demonstrates the CxList.FindByReturnType() method.
 The input source code is:
public class a
{
       int bla()
       {
              int b, a = 5;
               if (a == 33)
                       b = 6;
           return b;
      }
}
result = All.FindByReturnType ("int");
 the result would be -
      1 items found:
             bla() (in int bla())
```

### Version Information

Supported from v1.8.1

## 5.84 CxList.FindByShortName Method (string)

Returns a CxList which is a subset of this instance and its elements are the ones which their short name is the specified string.

### Syntax

```
CxQL
public CxList FindByShortName(string Name)
```

**Parameters**

**Name**

The short name of the objects to look for. Prefix and postfix wildcard (*) are supported.

**Return Value**

A subset of this instance and its elements are the ones which their name is the specified string.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByShortName method.

```
CxQL
```

```
 This example demonstrates the CxList.FindByShortName() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();

result = All.FindByShortName("Method");

 the result would be -
      1 item found:
           Method ( in b = a.Method() )
```

### Version Information

Supported from v1.8.1

## 5.85 CxList.FindByShortName Method (string, bool)

Returns a CxList which is a subset of this instance and its elements are the ones which their short name is the specified string, according to the specified comparison criteria.

### Syntax

CxQL
```
public CxList FindByShortName(string ShortName, bool caseSensitive)
```

**Parameters**

**ShortName**

Contains the short name of the objects. Prefix and postfix wildcard (*) are supported.

**caseSensitive**

Boolean which indicates to the search to be (or not) case sensitive.

**Return Value**

A subset of this instance and its elements are the ones which their short name is the specified string, according to the specified comparison criteria. Where the caseSensitive value can be true for case sensitive and false for case insensitive.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByShortName method.

CxQL
```

 This example demonstrates the CxList.FindByShortName() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
```

```
result = All.FindByShortName("method",true);
 the result would be -
      0 items found

result = All.FindByShortName("method", false);

 the result would be -
      1 item found:
             a.Method (in b = a.Method() )
```

### Version Information

Supported from v1.8.1

## 5.86 CxList.FindByShortNames Method (List<string>)

Returns a CxList which is a subset of this instance and its elements are the ones which their short name is the specified list of strings.

### Syntax

```
CxQL
public CxList FindByShortNames(List<string> nodeNames)
```

**Parameters**

**nodeNames**

The short names of the objects to look for. Prefix and postfix wildcard (*) are supported.

**Return Value**

A subset of this instance and its elements are the ones which their name listed in specified list of strings.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0). Works afficient if wildcard not present.

### Example

The following code example shows how you can use the FindByShortNames method.

```
CxQL

 This example demonstrates the CxList.FindByShortNames() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
 c = a.Method1()

result = All.FindByShortNames(new List<string> {"Method","Method1"});

 the result would be -
      2 item found:
             Method  ( in b = a.Method() )
             Method1 ( in c = a.Method1() )
```

### Version Information

Supported from v7.1.8

## 5.87 CxList.FindByShortNames Method (string[])

Returns a CxList which is a subset of this instance and its elements are the ones which their short name is the specified array of strings.

### Syntax

```
CxQL
public CxList FindByShortNames(string[] nodeNames)
```

**Parameters**

**nodeNames**

The short names of the objects to look for. Prefix and postfix wildcard (*) are supported.

**Return Value**

A subset of this instance and its elements are the ones which their name listed in specified array of strings.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0). Works afficient if wildcard not present.

### Example

The following code example shows how you can use the FindByShortNames method.

```
CxQL

 This example demonstrates the CxList.FindByShortNames() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
 c = a.Method1()

result = All.FindByShortNames(new string[] {"Method",”Method1”});

the result would be -
      2 item found:
             Method  ( in b = a.Method() )
             Method1 ( in c = a.Method1() )
```

### Version Information

Supported from v7.1.8

## 5.88 CxList.FindByShortNames Method (List<string>, bool)

Returns a CxList which is a subset of this instance and its elements are the ones which their short name is the specified string, according to the specified comparison criteria.

### Syntax

```
CxQL
public CxList FindByShortNames(List<string> nodeNames, bool caseSensitive)
```

**Parameters**

**nodeNames**

Contains the short name of the objects. Prefix and postfix wildcard (*) are supported.

**caseSensitive**

Boolean which indicates to the search to be (or not) case sensitive.

**Return Value**

A subset of this instance and its elements are the ones which their short name is the specified string, according to the specified comparison criteria. Where the caseSensitive value can be true for case sensitive and false for case insensitive.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0). Works afficient if wildcard not present.

### Example

The following code example shows how you can use the FindByShortName method.

```
CxQL

 This example demonstrates the CxList.FindByShortNames() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
 c = a.method1();

result = All.FindByShortNames(new List<string> {"method",”Method1”}, true);
 the result would be -
      1 items found
             c = a.method1();

result = All.FindByShortNames(new List<string> {"method",”Method1”},false);

 the result would be -
      2 item found:
             a.Method  (in b = a.Method() )
             a.method1 (in c = a.method1 () )
```

### Version Information

Supported from v7.1.8

## 5.89 CxList.FindByShortNames Method (string[], bool)

Returns a CxList which is a subset of this instance and its elements are the ones which their short name is the specified string, according to the specified comparison criteria.

### Syntax

```
CxQL
public CxList FindByShortNames(string[] nodeNames, bool caseSensitive)
```

**Parameters**

**nodeNames**

Contains the short name of the objects. Prefix and postfix wildcard (*) are supported.

**caseSensitive**

Boolean which indicates to the search to be (or not) case sensitive.

**Return Value**

A subset of this instance and its elements are the ones which their short name is the specified string, according to the specified comparison criteria. Where the caseSensitive value can be true for case sensitive and false for case insensitive.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0). Works afficient if wildcard not present.

### Example

The following code example shows how you can use the FindByShortName method.

```
CxQL

 This example demonstrates the CxList.FindByShortNames() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
 c = a.method1();

result = All.FindByShortNames(new string[] {"method","Method1"}, true);
 the result would be -
      1 items found
             c = a.method1();

result = All.FindByShortNames(new string[] {"method","Method1"},false);
```

```
 the result would be -
      2 item found:
             a.Method  (in b = a.Method() )
             a.method1 (in c = a.method1 () )
```

### Version Information

Supported from v7.1.8

## 5.90 CxList.FindByShortName Method (CxList)

Returns a CxList which is a subset of this instance and its elements are the ones which their short name is the specified string.

### Syntax

```
CxQL
public CxList FindByShortName(CxList nodesList)
```

**Parameters**

**nodesList**

The short name of the objects to look for. Prefix and postfix wildcard (*) are supported.

**Return Value**

A subset of this instance and its elements are the ones which their name is the specified string.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByShortName method.

```
CxQL

 This example demonstrates the CxList.FindByShortName() method.
 The input source code is:

    class Program
    {
        static void Main(string[] args)
        {
              Customer c = new customer();
        }
    }
class Customer{}
class User{}
CxList classes = All.FindByType(typeof(ClassDecl));
CxList types = All.FindByType(typeof(TypeRef));
CxList classesWithInstances = classes - classes.FindByShortName(types);

 the result would be -
      3 item found:
            Customer ( in class Customer{})
             Program ( in class Program)
```

```
              User ( in class User{})
```

### Version Information

Supported from v1.8.1

## 5.91 CxList.FindByShortName Method (CxList, bool)

Returns a CxList which is a subset of this instance and its elements are the ones which their short name is the specified string, according to the specified comparison criteria.

### Syntax

```
CxQL
public CxList FindByShortName(CxList nodesList, bool caseSensitive)
```

**Parameters**

**nodesList**

Contains the short name of the objects. Prefix and postfix wildcard (*) are supported.

**caseSensitive**

Boolean which indicates to the search to be (or not) case sensitive.

**Return Value**

A subset of this instance and its elements are the ones which their short name is the specified string, according to the specified comparison criteria. Where the csaeSensitive value can be true for case sensitive and false for case insensitive.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByShortName method.

```
CxQL

 This example demonstrates the CxList.FindByShortName() method.
 The input source code is:

    class Program
    {
        static void Main(string[] args)
        {
              Customer c = new customer();
        }
    }
class Customer{}
class User{}

CxList classes = All.FindByType(typeof(ClassDecl));
CxList types = All.FindByType(typeof(TypeRef));
CxList classesWithInstances = classes - classes.FindByShortName(types, true);

 the result would be -
```

```
      the same as FindByShortName(CxList nodesList)

CxList classes = All.FindByType(typeof(ClassDecl));
CxList types = All.FindByType(typeof(TypeRef));
CxList classesWithInstances = classes - classes.FindByShortName(types, false);

 the result would be -
      2 item found:
            Program ( in class Program)
              User ( in class User{})
```

### Version Information

Supported from v1.8.1

## 5.92 CxList.FindByTypeModifiers Method (TypeSignednessModifiers, TypeSizeModifiers)

Returns a CxList which is a subset of this instance and its elements are of the specified type modifiers of code element.

### Syntax

```
CxQL
public CxList FindByTypeModifiers(TypeSignednessModifiers TypeSignedness,
TypeSizeModifiers TypeSize)
```

**Parameters**

**TypeSignedness**

The type of the objects to be found. It can receive the following alternative values:

- TypeSignednessModifiers.Unknown
- TypeSignednessModifiers.Signed
- TypeSignednessModifiers.Unsigned

**TypeSize**

The type of the objects to be found. It can receive the following alternative values:

- TypeSignednessModifiers.Default
- TypeSignednessModifiers.Short
- TypeSignednessModifiers.Long
- TypeSignednessModifiers.LongLong

**Return Value**

A subset of this instance which elements type contains both modifiers provided.

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByType method.

```
CxQL

 This example demonstrates the CxList.FindByTypeModifers() method.
 The input source code is:
 unsigned long int a;
 int b;
 b = a++;
```

```
result = All.FindByTypeModifiers (TypeSignednessModifiers.Unsigned,
TypeSizeModifiers.Long);
 the result would be -
      3 items found:
             int (in unsigned long int a)
             a (in unsigned long int a)
             a (in b = a++)
```

### Version Information

Supported from v8.8.0

## 5.93 CxList.FindByTypeModifiers Method (TypeSignednessModifiers)

Returns a CxList which is a subset of this instance and its elements are of the specified type modifiers of code element.

### Syntax

```
CxQL
public CxList FindByTypeModifiers(TypeSignednessModifiers TypeSignedness,
TypeSizeModifiers TypeSize)
```

**Parameters**

**TypeSignedness**

The type of the objects to be found. It can receive the following alternative values:

- TypeSignednessModifiers.Unknown
- TypeSignednessModifiers.Signed
- TypeSignednessModifiers.Unsigned

**Return Value**

A subset of this instance which elements type contains the modifier provided.

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByType method.

```
CxQL

 This example demonstrates the CxList.FindByTypeModifers() method.
 The input source code is:
 unsigned long int a;
 int b;
 b = a++;

result = All.FindByTypeModifiers (TypeSignednessModifiers.Unsigned);
 the result would be -
      3 items found:
             int (in unsigned long int a)
             a (in unsigned long int a)
             a (in b = a++)
```

### Version Information

Supported from v8.8.0

## 5.94 CxList.FindByTypeModifiers Method (TypeSizeModifiers)

Returns a CxList which is a subset of this instance and its elements are of the specified type modifiers of code element.

### Syntax

```
CxQL
public CxList FindByTypeModifiers(TypeSignednessModifiers TypeSignedness,
TypeSizeModifiers TypeSize)
```

**Parameters**

**TypeSize**

The type of the objects to be found. It can receive the following alternative values:

- TypeSignednessModifiers.Default
- TypeSignednessModifiers.Short
- TypeSignednessModifiers.Long
- TypeSignednessModifiers.LongLong

**Return Value**

A subset of this instance which elements type contains the modifier provided.

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByType method.

```
CxQL

 This example demonstrates the CxList.FindByTypeModifers() method.
 The input source code is:
 unsigned long int a;
 int b;
 b = a++;

result = All.FindByTypeModifiers (TypeSizeModifiers.Long);
 the result would be -
      3 items found:
             int (in unsigned long int a)
             a (in unsigned long int a)
             a (in b = a++)
```

### Version Information

Supported from v8.8.0

## 5.95 CxList.FindByType Method (Type)

Returns a CxList which is a subset of this instance and its elements are of the specified type of code element.

### Syntax

```
CxQL
public CxList FindByType(Type TypeName)
```

**Parameters**

**TypeName**

The type of the objects to be found

**Return Value**

A subset of this instance and its elements are of the specified type of code element.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByType method.

```
CxQL

 This example demonstrates the CxList.FindByType() method.
 The input source code is:
 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();

result = All.FindByType (typeof(MemberAccess));
 the result would be -
      2 items found:
             a.DataMember (in a.DataMember = 3)
             a.Method (in b = a.Method())
```

### Version Information

Supported from v1.8.1

## 5.96 CxList.FindByType Method (string)

Returns a CxList which is a subset of this instance and its elements are of the specified type.

### Syntax

```
CxQL
public CxList FindByType(String Type)
```

**Parameters**

**Type**

The type of the objects to be found

**Return Value**

A subset of this instance and its elements are of the specified type.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByType method.

```
CxQL

 This example demonstrates the CxList.FindByType() method.
 The input source code is:
 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();

result = All.FindByType ("MyClass");
 the result would be -
      3 items found:
             a (in MyClass a)
             a (in a.DataMember = 3)
             a (in b = a.Method())
```

### Version Information

Supported from v1.8.1

## 5.97 CxList.FindByType Method (string, bool)

Returns a CxList which is a subset of this instance and its elements are of the specified type.

### Syntax

```
CxQL
public CxList FindByType(String Type, bool CaseSensitive)
```

**Parameters**

**Type**

The type of the objects to be found

**CaseSensitive**

Ignore case true/false

**Return Value**

A subset of this instance and its elements are of the specified type.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByType method.

```
CxQL

 This example demonstrates the CxList.FindByType() method.

 The input source code is:
 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();

result = All.FindByType ("MyClass",true);
 the result would be -
      3 items found:
             a (in MyClass a)
             a (in a.DataMember = 3)
             a (in b = a.Method())
```

### Version Information

Supported from v1.8.1

## 5.98 CxList.FindByTypes Method (params string[])

Returns a CxList which is a subset of this instance and its elements are of the specified type.

### Syntax

```
CxQL
public CxList FindByType(params String[] Types)
```

**Parameters**

**Types**

The types of the objects to be found

**Return Value**

A subset of this instance and its elements are of the specified types.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByType method.

```
CxQL

 This example demonstrates the CxList.FindByType() method.
 The input source code is:
 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();

String[] arr = new String[]{"MyClass","int"};
result = All.FindByTypes(arr);
the result would be -
      6 items found:
             a (in MyClass a)
             a (in a.DataMember = 3)
             a (in b = a.Method())
             b (in int b)
              b (in b = a.Method())
             MyClass (in MyClass a)
```

### Version Information

Supported from v1.8.1

## 5.99 CxList.FindByTypes Method (string[], bool)

Returns a CxList which is a subset of this instance and its elements are of the specified type.

### Syntax

```
CxQL
public CxList FindByType(String[] Types, bool caseSensitive)
```

**Parameters**

**Types**

The types of the objects to be found

**CaseSensitive**

Ignore case true/false

**Return Value**

A subset of this instance and its elements are of the specified types.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByType method.

```
CxQL

 This example demonstrates the CxList.FindByType() method.
 The input source code is:

 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
String[] arr = new String[]{"MyClass","int"};
result = All.FindByTypes(arr,false);
the result would be -
      6 items found:
             a (in MyClass a)
             a (in a.DataMember = 3)
             a (in b = a.Method())
             b (in int b)
              b (in b = a.Method())
             MyClass (in MyClass a)
```

### Version Information

Supported from v7.1.8

## 5.100 CxList.FindByTypes Method (params Type[])

Returns a CxList which is a subset of this instance and its elements are of the specified list of types.

### Syntax

```
CxQL
public CxList FindByTypes(param Type[] types)
```

**Parameters**

**Types**

The list of types objects to be found

**Return Value**

An  subset of this instance and its elements are of the specified list of types.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByTypes method.

```
CxQL
 This example demonstrates the CxList.FindByTypes(params Type[]) method.
 The input source code is:
 MyClass a;
 a.DataMember = 3;

result = All.FindByTypes(typeof(MemberAccess), typeof(AssignExpr));
 the result would be -
   2 items found:
     a.DataMember (in a.DataMember = 3) // MemberAccess
      = (in a.DataMember = 3) // AssignExpr
```

### Version Information

Supported from v9.4.0

## 5.101 CxList.FindByType<T> Method ()

Returns a CxList which is a subset of this instance and its elements are of the given type of code element.

### Syntax

```
CxQL
public CxList FindByType<T>()
```

**Parameters**

**Return Value**

A subset of this instance and its elements are of the given type of code element.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByType method.

```
CxQL
 This example demonstrates the CxList.FindByType<T>() method.
 The input source code is:
 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();

result = All.FindByType<MemberAccess>();
 the result would be -
   2 items found:
     a.DataMember (in a.DataMember = 3)
     a.Method (in b = a.Method())
```

### Version Information

Supported from v9.4

## 5.102 CxList.FindByReturnTypes Method (params string[])

Returns a CxList which is a subset of this instance and its elements are of the specified list of types.

### Syntax

```
CxQL
public CxList FindByReturnType(String[] Types, bool stripPointerTypeRefFromRetur
nType = true)
```

**Parameters**

**Types**

The types of the objects to be found

**stripPointerTypeRefFromReturnType**

true – the result will include methods that return Type* as well as methods that return Type

**Return Value**

A subset of this instance and its elements are of the specified return type.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByReturnTypes method.

```
CxQL
 This example demonstrates the CxList.FindByReturnTypes() method.
 The input source code is:
public class a
{
int bla()
{
 int b, a = 5;
 if (a == 33)
 b = 6;
```

```
    return b;
      }

      boolean test()
      {
    return true;
      }

}
String[] types = new String[]{"int", "boolean"};
result = All.FindByReturnTypes(types);
 the result would be -
   2 items found:
     bla() (in int bla())
      test() (in boolean test())
```

### Version Information

Supported from v9.4.0

## 5.103 CxList.FindByReturnTypes Method (params string[], bool stripPointerTypeRefFromReturnType = true)

Returns a CxList which is a subset of this instance and its elements are of the specified list of types.

### Syntax

```
CxQL
public CxList FindByReturnType(String[] Types, bool stripPointerTypeRefFromRetur
nType = true)
```

**Parameters**

**Types**

The types of the objects to be found

**stripPointerTypeRefFromReturnType**

true – the result will include methods that return Type* as well as methods that return Type

**Return Value**

A subset of this instance and its elements are of the specified return type.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the FindByReturnTypes method.

```
CxQL
 This example demonstrates the CxList.FindByReturnTypes() method.
 The input source code is:
public class a
{
int bla()
{
 int b, a = 5;
 if (a == 33)
 b = 6;
    return b;
      }

      boolean test()
      {
    return true;
      }

}
String[] types = new String[]{"int", "boolean"};
result = All.FindByReturnTypes(types);
 the result would be -
   2 items found:
     bla() (in int bla())
      test() (in boolean test())
```

### Version Information

Supported from v9.4.0

## 5.104 CxList.FindDefinition Method (CxList)

Returns a CxList which is a subset of "this" instance, with elements that are the definition locations of the first element in the given CxList.

### Syntax

```
CxQL
public CxList FindDefinition(CxList items)
```

**Parameters**

**Items**

Items whose definition to be found.

**Return Value**

A subset of "this" instance, with elements that are the definition locations of the first element in the specified CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindDefinition() method.
 The input source code is:

 int b, a = 5;
 if (a > 3)
```

```
      b = a;

result = All.FindDefinition(All.FindByName("*b*"));

 The result would consist of 1 item:
             b (in int b, a = 5)
```

### Version Information

Supported from v1.8.1

## 5.105 CxList.FindInitialization Method (CxList)

Returns a CxList which is a subset of “this” instance and the elements are the initialization values of the elements from the given CxList.

### Syntax

```
CxQL
public CxList FindInitialization(CxList declarators)
```

**Parameters**

**Declarators**

A CxList of declarators.

**Return Value**

A subset of “this” instance whose elements are the initialization values of the given CxList elements.

Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.FindInitialization() method.
 The input source code is:

 int b = 5;

CxList declarators = All.FindByType(typeof(Declarator));
result = All.FindInitialization(declarators);

 The result would consist of 1 item:
             5
```

### Version Information

Supported from v1.8.1

## 5.106 CxList.FindInScope Method (CxList, CxList)

Returns a CxList which is a subset of “this” instance with the elements inside the scopes defined by the start nodes CxList and the end nodes CxList.

### Syntax

```
CxQL
public CxList FindInScope(CxList StartNodes, CxList EndNodes)
```

**Parameters**

**StartNodes**

the nodes that define the start of the scope.

**EndNodes**

the nodes that define the end of the scope.

**Return Value**

all nodes found inside the given scope.

### Example

```
This example demonstrates the CxList.FindInScope(CxList StartNodes, CxList
EndNodes) method.
 The input source code is:
       @{ Html.BeginForm(); } //Defines the start of the scope
       @Html.AntiForgeryToken();
       @{ Html.EndForm(); } //Defines the end of the scope

CxList BeginFormMethods = All.FindByShortName("BeginForm");
CxList EndFormMethods = All.FindByShortName("EndForm");
result = All.FindInScope(BeginFormMethods, EndFormMethods);

the result would consist of 1 item:
            AntiForgeryToken(@Html.AntiForgeryToken();)
The purpose of the query is to find anything that is contained inside a specific
scope.
```

### Version Information

Supported from 8.5.0

## 5.107 CxList.FindSubList Method (int, bool)

Returns a CxList which is a subset of “this”. The following interface provides the ability to extract N elements from the CxList . **The main purpose of this interface is NOT use with member “data” of CxList.**

### Syntax

```
CxQL
public CxList FindSubList(int count, bool fromStart)
```

**Parameters**

**count**

number of nodes to extract.

**fromStart**

If the parameter value is true it means get elements from start of the list

If the parameter value is false it means get elements from end of the list

.

**Return Value**

Requested count elements.

### Example

```
Example 1: Get last element and add it to result
// Current Implementation
```

```
CSharpGraph secondParam = secondParameterOfSetHeader.data.GetByIndex(secondParam
eterOfSetHeader.Count-1) as CSharpGraph;
result.Add(secondParam.NodeId, secondParam);

// New Implementation
result.Add(secondParameterOfSetHeader.FindSubList (1,false));

Example 2: Get first and second element. Assume that all elementas are
BinaryExpr.

// Current Implementation
// currently not supporting logical conditions with more than two sons
if (curSons.Count >= 2)
{
     continue;
}

BinaryExpr firstOp = curSons.data.GetByIndex(0) as BinaryExpr;
BinaryExpr secondOp = curSons.data.GetByIndex(1) as BinaryExpr;


// New Implementation
// currently not supporting logical conditions with more than two sons
if (curSons.Count >= 2)
{
     continue;
}

// get first 2 elements of the list
CxList secondOpTemp  = curSons. FindSubList (2,true));
//get first one
BinaryExpr firstOp =
       secondOpTemp.FindSubList(1,true)).TryGetCSharpGraph<BinaryExpr>();

//get last one (second of original list)
BinaryExpr secondOp =
       secondOpTemp.FindSubList(1,false)).TryGetCSharpGraph<BinaryExpr>();
```

### Version Information

Supported from 9.2.0

## 5.108 CxList.GetAncOfType Method (Type)

Returns a CxList with all the elements that are CxDOM first ancestor of the calling CxList and which are of type t. First ancestor means that it searches upward in the CxDOM graph until the first ancestor matching the condition (type t), and NOT that it searches only for fathers

### Syntax

```
CxQL
public CxList GetAncOfType(Type t)
```

**Parameters**

The type of DOM objetct the methods looks for

**Return Value**

Returns a CxList with all the CxDOM elements of type t, which are first ancestor, of some element in the calling CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).
This command does not return a subset of the CxList, but a subset of All.

### Example

The following code example shows how you can use the GetAncOfType method.

```
CxQL

 This example demonstrates the CxList.GetAncOfType() method.
 The input source code is:

 if (a>b)
 {
      c=100;
 }
 else
 {
      if(a<100)
      {
       d=200;
      }
 }

result = All.FindByName("d"). GetAncOfType(typeof(IfStmt));
 the result would be -
      1 item found:
      if (in  if(a<100))
```

### Version Information

Supported from  v2.0.5

## 5.109 CxList.GetAncOfType<Type> Method () where T : CSharpGraph

Returns a CxList with all the elements that are CxDOM first ancestor of the calling CxList and which are of type “T”. First ancestor means that it searches upward in the CxDOM graph until the first ancestor matching the generic type “T”, and NOT that it searches only for fathers.

### Syntax

```
CxQL
public CxList GetAncOfType<T>()
```

**Parameters**

<T> the type to cast the DOM object to (must inherit from CSharpGraph)

**Return Value**

Returns a CxList with all the CxDOM elements of type “T”, which are first ancestor, of some element in the calling CxList.

### Remarks

The return value may be empty (Count = 0).
This command does not return a subset of the CxList, but a subset of All.

### Example

The following code example shows how you can use the GetAncOfType method.

```
CxQL
 This example demonstrates the CxList.GetAncOfType<CSharpGraph>() method.
The input source code is:
 if (a>b)
 {
   c=100;
 }
 else
 {
   if(a<100)
   {
   d=200;
   }
 }

result = All.FindByName("d").GetAncOfType<IfStmt>();
 the result would be -
   1 item found:
  if (inif(a<100))
```

### Version Information

Supported from v9.4.0

## 5.110 CxList.GetArrayOfNodeIds Method ()

Returns an ArrayList which is a set of all elements IDs All this CxList.

### Syntax

```
CxQL
public ArrayList GetArrayOfNodeIds()
```

**Parameters**

None

**Return Value**

ArrayList which is a set of all elements IDs All this CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Example

The following code example shows how you can use the FindByReturnType method.

```
CxQL

 This example demonstrates the CxList.GetArrayOfNodeIds() method.
 The input source code is:
```

```
public class a
{
void foo(){
 MyClass a;
 int b;
 a.DataMember = 3;
 b = a.Method();
 }
}
CxList ls = All;
foreach(int NodeId in ls.GetArrayOfNodeIds())
{
       if(NodeId !=1)
       {
              result = All.FindById(NodeId);
       }
}
```

### Version Information

Supported from : v1.8.1

## 5.111 CxList.GetBlocksOfIfStatements Method (bool)

Returns a CxList with all the true/false blocks of the if statements inside the calling CxList according the provided boolean parameter. The true block of an if statement is the block which is ran if the condition is verified, whereas the false block is the else block (if it exists).

### Syntax

```
CxQL
public CxList GetBlocksOfIfStatements(Boolean block)
```

**Parameters**

A Boolean describing wether you want to retrieve the true or false blocks of the if statements inside the calling CxList.

**Return Value**

Returns a CxList containing all the true/false blocks of the if statements contained in the calling CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).
This command does not return a subset of the CxList, but a subset of All.

### Example

The following code example shows how you can use the GetBlocksOfIfStatements method.

```
CxQL

 This example demonstrates the CxList.GetBlocksOfIfStatements(block) method.
 The input source code is:

 int a = 5;
 if (a > 3)
 {
       a = 4;
 }
 else if(a > 2)
 {
       a = 2;
 }
 else
 {
       a = 8;
 }

result = All.GetBlocksOfIfStatements(true);

 The result would consist of 2 items: the braces after if(a>3) (true block of
the if), and the braces after the else if (true block of the if).

 result = All.GetBlocksOfIfStatements(false);

 This query ran on the same sample code woud return the if(a>2) (false block of
the else in else if), and the braces after the else.
```

### Version Information

Supported from v8.5.0

## 5.112 CxList.GetBlocksOfIterationStatements Method ()

Returns a CxList with the blocks of all the iterations contained in the calling CxList.

### Syntax

```
CxQL
public CxList GetBranchesOfTernaryExpression(Boolean branch)
```

**Parameters**

None.

**Return Value**

Returns a CxList containing all the blocks of the iterations in the calling CxList.

### Remarks

The return value may be empty (Count = 0).
This command does not return a subset of the CxList, but a subset of All.

### Example

The following code example shows how you can use the GetBlocksOfIterationStatements method.

```
CxQL

 This example demonstrates the CxList.GetBlocksOfIterationStatements() method.
 The input source code is:

 while(condition)
 {
       a = 1;
 }
 for(int i=0; i<10; i++)
 {
       // code
 }
```

```
result = All.GetBlocksOfIterationStatements();

 The result would consist of 2 item: the 2 opening braces in the code.
```

### Version Information

Supported from v8.5.0

## 5.113 CxList.GetBranchesOfTernaryExpressions Method (bool)

Returns a CxList with all the true/false branches of the ternary expressions inside the calling CxList according the provided boolean parameter. The true branch of a ternary expression is the value given if the condition is true, whereas the false block is the value given if the condition is false.

### Syntax

CxQL

```
public CxList GetBranchesOfTernaryExpression(Boolean branch)
```

**Parameters**

A boolean describing wether you want to retrieve the true or false branches of the ternary expressions inside the calling CxList.

**Return Value**

Returns a CxList containing all the true/false branches of the ternary expressions in the calling CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).
This command does not return a subset of the CxList, but a subset of All.

### Example

The following code example shows how you can use the GetBranchesOfTernaryExpressions method.

CxQL

```
 This example demonstrates the CxList.GetBranchesOfTernaryExpressions(branch)
method.
 The input source code is:

 int a = someCondition ? 1:0;

result = All.GetBranchesOfTernaryExpressions(true);

 The result would consist of 1 item: the number 1.

result = All.GetBranchesOfTernaryExpressions(false);

 The result would consist of 1 item: the number 0.
```

### Version Information

Supported from v8.5.0

## 5.114 CxList.GetByAncs Method (CxList)

Returns all elements in this instance that is a CxDOM descendant of an element of the parameter.

### Syntax

```
CxQL
public CxList GetByAncs(CxList ancs)
```

**Parameters**

**ancs**

The Ancestors whose descendants are to be returned

**Return Value**

Returns all elements in this instance that descends any of the elements in the parameter

### Example

The following code example shows how you can use the GetByAncs method.

```
CxQL

 This example demonstrates the CxList.GetByAncs() method.
 The input source code is:
 public notmuch (boolean tf)
{
      boolean localboolean = tf;
}

result = All.GetByAncs(All.FindByName("notmuch"));

 6 items found:
notmuch
boolean (in Boolean tf)
tf
boolean
localboolean
=
tf (in localoolean=tf)
```

### Version Information

Supported from v2.0.5

## 5.115 CxList.GetByBinaryOperator Method (BinaryOperator)

Returns a CxList which is a subset of this instance and its elements are binary expressions with a given binary operator.

### Syntax

```
CxQL
public CxList GetByBinaryOperator(BinaryOperator opr)
```

**Parameters**

**opt**

Enum type of binary operators.

**Return Value**

A subset of this instance with binary expressions which have a given binary operator.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the GetByBinaryOperator method.

```
CxQL

 This example demonstrates the CxList.GetByBinaryOperator() method.
 The input source code is:

 int i;
 if(i < 1)
       ...

result = All.GetByBinaryOperator(BinaryOperator.LessThan);

 the result would be -
      1 item found:
             <
```

### Version Information

Supported from  v1.8.1

## 5.116 CxList.GetByClass Method (CxList)

Returns all elements in “this” instance that belong to any of the classes in the parameter.

### Syntax

```
CxQL
public int GetByClass(CxList classes)
```

**Parameters**

classes

The classes whose elements to be returned

**Return Value**

Returns all elements in this instance that belong to any of the classes in the parameter

### Example

```
CxQL

 This example demonstrates the CxList.GetByClass() method.
 The input source code is:
 class cl1
 {
      void foo()
      {
             int a = 3;
             int b = 5;
      }
 }
 class cl2
 {
      void foo2()
      {
             int c = 3;
      }
 }

result = All.GetByClass(All.FindByName("*.cl1")).FindByName("3");

 The result would consist of 1 item found:
      3 (in int a = 3)
 Notice that 3 (in int c = 3) doesn't appear in the results, since it is not  in
the "cl1" class
```

## 5.117 CxList.GetByMethod Method (CxList)

Returns all elements in this instance that belong to any of the methods  in the parameter

### Syntax

```
CxQL
public int GetByMethod(CxList methods)
```

**Parameters**

**methods**

The methods  whose elemets to be returned

**Return Value**

Returns all elements in this instance that belong to any of the methods in the parameter

### Example

The following code example shows how you can use the GetByMethod method.

```
CxQL

 This example demonstrates the CxList.GetByMethod() method.
 The input source code is:
 class cl1
 {
      void foo()
      {
             int a = 3;
             int b = 5;
      }
      void foo2()
      {
             int c = 3;
      }
 }

result = All.GetByMethod(All.FindByName("foo2")).FindByName("3");

 1 item found:
      3 (in int c = 3)
 Notice that 3 (in int a = 3) doesn't appear in the results, since it is not  in
the "foo2" method
```

### Version Information

Supported from v2.0.5

## 5.118 CxList.GetClass Method (CxList)

Returns the classes of this instance containing the objects in the parameter.

### Syntax

```
CxQL
public CxList GetClass(CxList elements)
```

**Parameters**

**elements**

The elements whose classes to be returned

**Return Value**

Returns the classes of this instance containing the objects in the parameter.

### Example

The following code example shows how you can use the GetClass method.

```
CxQL

 This example demonstrates the CxList.GetClass() method.
 The input source code is:
 class cl1
 {
      void foo()
      {
             int a = 3;
             int b = 5;
      }
 }

result = All.GetClass(All.FindByName ("5"));

 1 item found:
      cl1 (in class cl1)
```

### Version Information

Supported from v2.0.5

## 5.119 CxList.GetCxListByPath Method ()

Create enumerator on CxList that enumerate on all existing paths.

### Syntax

```
CxQL
public IEnumerable<CxList> GetCxListByPath()
```

**Parameters**

**No parameters**

## Exceptions

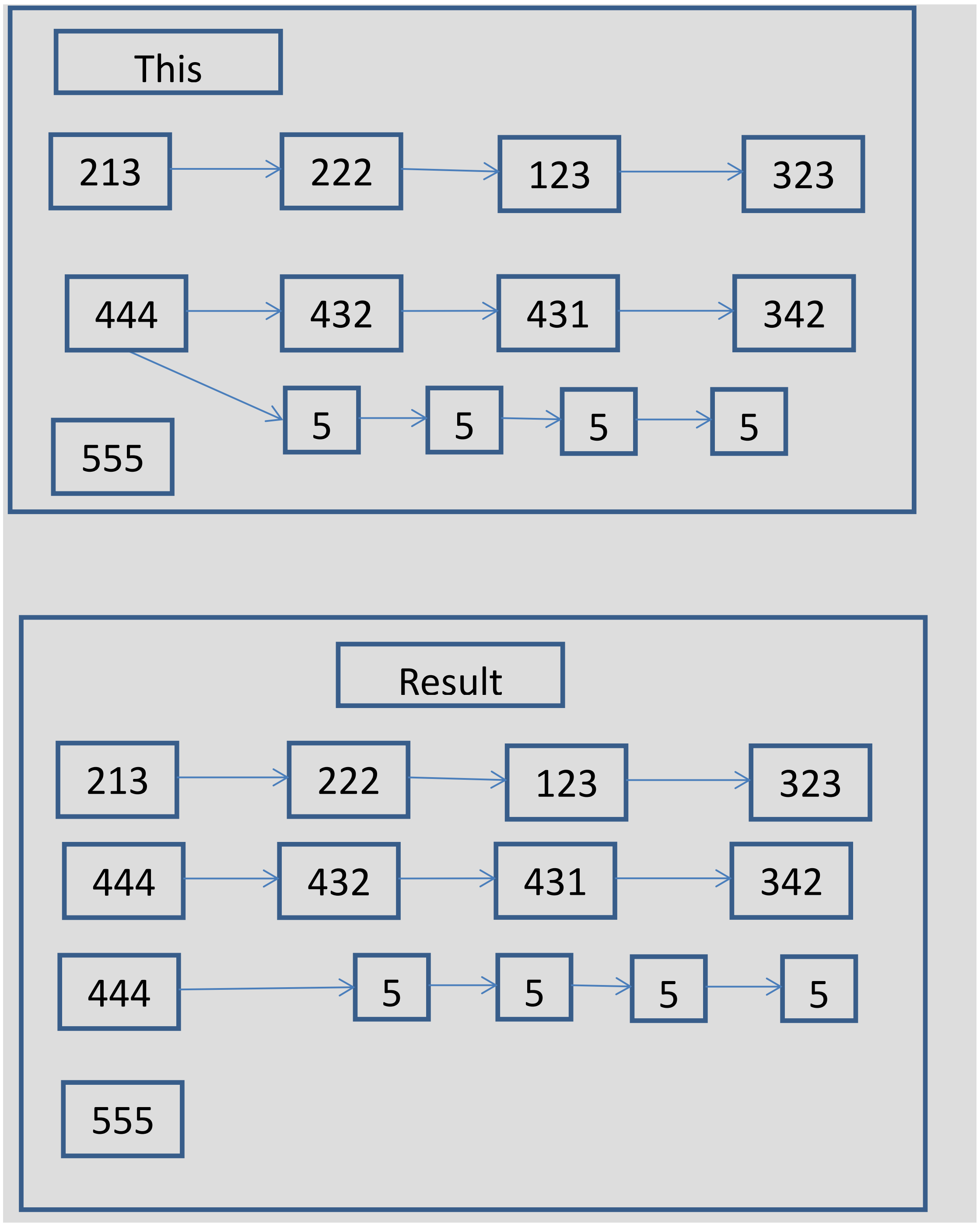
| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

## Remarks

None

## Example

```
CxQL

 This example demonstrates the IEnumerable<CxList> GetCxListByPath() method.

foreach (CxList thisCxList in this.GetCxListByPath())
{
   // thisCxList shall include one node and one path. If in “this” exists nodes without
   // pathes than thisCxList will have only one node.
}
```



### Version Information

Supported from v7.1.3

## 5.120 CxList.GetEnumerator Method ()

Return IEnumerator of CxList.Data

### Syntax

```
CxQL
public IEnumerator GetEnumerator()
```

**Parameters**

none

**Return Value**

Enumerator of CxList.Data.

## Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

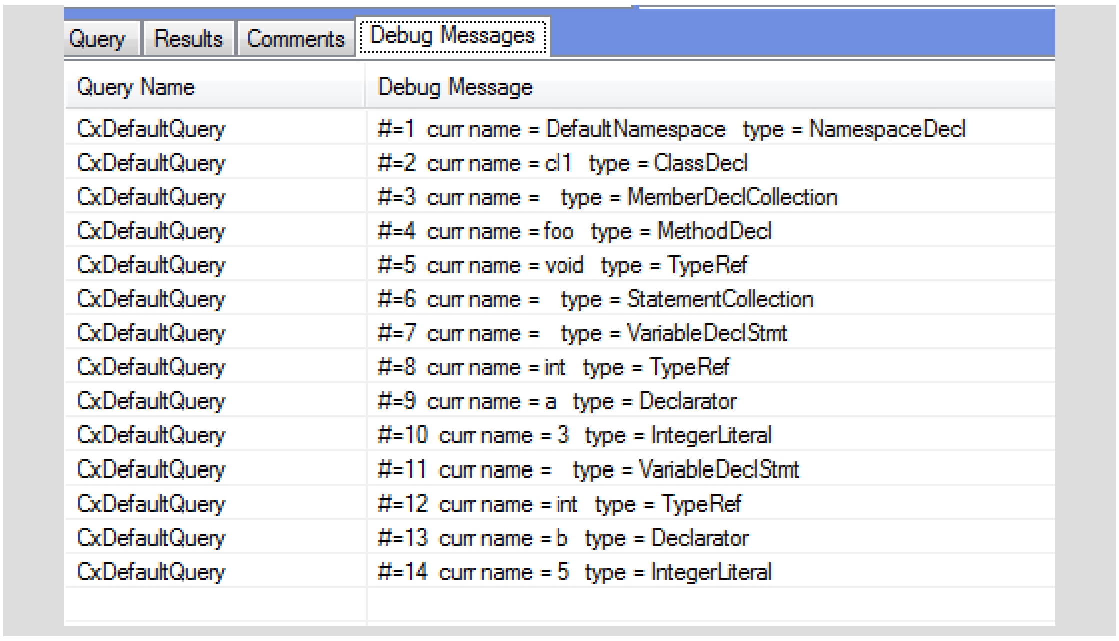
## Remarks

Not in use (deprecated). A simpler implementation is by:

```
foreach  (CxList cxItem in resultList)
{
   :
}
```

## Example

```
CxQL

 This example demonstrates the GetEnumerator, cxLog.WriteDebugMessage and
GetFirstGraph method.
 The input source code is:

class cl1
 {
      void foo()
      {
             int a = 3;
             int b = 5;
      }
 }

IEnumerator ieNum = All.GetEnumerator();
bool finish = false;
int i = 1;
while (!finish)
{
       if (!ieNum.MoveNext())
       {
              finish = true;
       }
       else
       {
              CxList curr = (CxList) ieNum.Current;
              if (curr.GetFirstGraph() != null)
              {
                     cxLog.WriteDebugMessage ("#=" + i.ToString() +
                     "   curr name = " + curr.GetName() + "    type = " +
                   curr.GetFirstGraph().GraphType.ToString());
                     i++;
              }
       }
}
   the result would be on DebugMessage tab in CxAudit program
```


Query | Results | Comments | Debug Messages

| Query Name | Debug Message |
|---|---|
| CxDefaultQuery | #=1 curr name = DefaultNamespace type = NamespaceDecl |
| CxDefaultQuery | #=2 curr name = cl1 type = ClassDecl |
| CxDefaultQuery | #=3 curr name = type = MemberDeclCollection |
| CxDefaultQuery | #=4 curr name = foo type = MethodDecl |
| CxDefaultQuery | #=5 curr name = void type = TypeRef |
| CxDefaultQuery | #=6 curr name = type = StatementCollection |
| CxDefaultQuery | #=7 curr name = type = VariableDeclStmt |
| CxDefaultQuery | #=8 curr name = int type = TypeRef |
| CxDefaultQuery | #=9 curr name = a type = Declarator |
| CxDefaultQuery | #=10 curr name = 3 type = IntegerLiteral |
| CxDefaultQuery | #=11 curr name = type = VariableDeclStmt |
| CxDefaultQuery | #=12 curr name = int type = TypeRef |
| CxDefaultQuery | #=13 curr name = b type = Declarator |
| CxDefaultQuery | #=14 curr name = 5 type = IntegerLiteral |

### Version Information

Supported from v1.8.1

## 5.121 CxList.GetFathers Method ()

Returns a CxList which contains the direct fathers of the elements of “this” instance.

### Syntax

```
CxQL
public CxList GetFathers ()
```

**Return Value**

A CxList which contains the direct fathers of the element of “this” instance.

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.GetFathers () method.
 The input source code is:

 int b, a = 5;
 if (a > 3)
      b = 6;

CxList six = All.FindByName("6");
result = six. GetFathers();

 the result would consist of 1 item found:
            =
```

## 5.122 CxList.GetFinallyClause Method (CxList)

Returns a CxList which is a subset of this instance and its elements are finally clauses of the specified CxList of try statements.

### Syntax

```
CxQL
public CxList GetFinallyClause (CxList TryList)
```

**Parameters**

**TryList**

CxList of try statements.

**Return Value**

A subset of this instance with finally clauses.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the GetFinallyClause method.

```
CxQL
 This example demonstrates the CxList.GetFinallyClause() method.
 The input source code is:


int j;
 try
 {
       int i = 0;
       j = 1 / i;
 }
 finally
 {
       j = 1;
 }

CxList Try = All.FindByType(typeof(TryCatchFinallyStmt));
result = All.GetFinallyClause(Try);
 the result would be -
      1 item found:
            finally
```

### Version Information

Supported from v1.8.1

## 5.123 CxList.GetFirstGraph Method ()

Returns a first data element in requested CxList. Using to get internal data of first object in requested CxList

### Syntax

```
CxQL
public CSharpGraph GetFirstGraph()
```

**Parameters**

none

**Return Value**

A first element in Data. If CxList empty return null.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

N/A

### Example

```
CxQL

 This example demonstrates the CxList.GetFirstGraph() method.
 The input source code is:

class cl1
 {
      void foo()
      {
             int a = 3;
             int b = 5;
      }
 }
result = All.FindByShortName("foo");
if (result.Count > 0)
       cxLog.WriteDebugMessage(result.GetFirstGraph().ShortName);

 the result would be on DebugMessage tab in CxAudit program
       foo
```

### Version Information

Supported from  v1.8.1

## 5.124 CxList.GetFollowingStatements Method ()

Returns a CxList of the statements that are directly following all the statements of the calling CxList and in the same statement collection.

### Syntax

```
CxQL
public CxList GetFollowingStatements()
```

**Parameters**

None.

**Return Value**

Returns a CxList containing all the statements that are following the statements in the calling CxList.

### Remarks

The return value may be empty (Count = 0).
This command does not return a subset of the CxList, but a subset of All.

### Example

The following code example shows how you can use the GetFollowingStatements method.

```
CxQL

 This example demonstrates the CxList.GetFollowingStatements() method.
 The input source code is:

 if (a > 3)
 {
       a = 4;
 }
 if(a != 4)
 {
       a = 0;
       b = 5;
 }
 c = 2;
 d = 3;

result = All.GetFollowingStatements();

 The result would consist of 2 items: the assignment b=5 (following a=0),
and d=3 (following c=2).
 The assignment a=4 doesn't have any following statement in its scope.
```

### Version Information

Supported from v8.5.0

## 5.125 CxList.GetMembersOfTarget Method ()

Returns a CxList with all found members of the specified target.

### Syntax

```
CxQL
public CxList GetMembersOfTarget()
```

**Return Value**

A CxList with members of a given target.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the GetMembersOfTarget method.

```
CxQL
```

```
 This example demonstrates the CxList. GetMembersOfTarget() method.
 The input source code is:

 StreamWriter sw = new StreamWriter();
 sw.Write("");

CxList sWriter = All.FindByType("StreamWriter");
result = sWriter. GetMembersOfTarget();

 the result would be -
      1 item found:
             Write
```

### Version Information

Supported from v1.8.1

## 5.126 CxList GetRightmostMember()

Returns a CxList with the rightmost members of the specified target.

### Syntax

```
CxQL
public CxList GetRightmostMember()
```

**Return Value**

A CxList with the rightmost members of a given target.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the GetRightmostMember method.

```
CxQL

 This example demonstrates the CxList. GetRightmostMember() method.
 The input source code is:

int i = foo().Bar().a.b;

CxList foo = All.FindByName("foo");
result = foo. GetRightmostMember();

 the result would be -
      1 item found:
             b
```

### Version Information

Supported from v8.0

## 5.127 CxList GetLeftmostTarget()

Returns a CxList with leftmost target of the specified member.

### Syntax

```
CxQL
public CxList GetLeftmostTarget()
```

**Return Value**

A CxList with the leftmost target of a given member.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the GetRightmostMember method.

```
CxQL

 This example demonstrates the CxList. GetLeftmostTarget() method.
 The input source code is:

int i = foo().Bar().a.b;

CxList b = All.FindByName("b");
result = b.LeftmostTarget();

 the result would be -
      1 item found:
             foo
```

### Version Information

Supported from v8.0

## 5.128 CxList.GetMembersWithTargets Method ()

Returns a CxList which is a subset of "this" instance with nodes that are part of a member/target pair (typical example: target.member) and have a direct target (i.e. they are the member).

### Syntax

```
CxQL
public CxList GetMembersWithTargets()
```

**Return Value**

Returns a CxList which is a subset of "this" instance with nodes that have a direct target.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the GetMembersWithTargets method.

```
CxQL

 This example demonstrates the CxList.GetMembersWithTargets() method.
 The input source code is:

 int num = 55;
 string Str = num.ToString().ToUpper().PadLeft(5, ' ');

CxList methods = All.FindByType(typeof(MethodInvokeExpr));
result = methods.GetMembersWithTargets();

 the result would be -
      3 items found:
            PadLeft, ToUpper, ToString in num.ToString().ToUpper().PadLeft(5, '
');
```

### Version Information

Supported from v1.8.1

## 5.129 CxList.GetMembersWithTargets Method (CxList)

Returns a CxList which is a subset of “this” instance with nodes that are part of a member/target pair (typical example: target.member) and have a direct target in the CxList paramemeter "targets".

### Syntax

```
CxQL
public CxList GetMembersWithTargets(CxList targets)
```

**Parameters**

targets - CxList of DOM objects which might be the target(s) of elements in "this"

**Return Value**

Returns a CxList which is a subset of “this” instance with nodes that have a direct target in **"targets"** parameter

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Remarks

The return value may be empty (Count = 0).

If targets is null – returns an empty CxList

### Example

The following code example shows how you can use the GetMembersWithTargets method.

```
CxQL

 This example demonstrates the CxList.GetMembersWithTargets() method.
 The input source code is:
```

```
 int num = 55;
 string Str = num.ToString().ToUpper().PadLeft(5, ' ');

CxList methods = All.FindByType(typeof(MethodInvokeExpr));
CxList num = All.FindByShortName("num");
result = methods.GetMembersWithTargets(num);

 the result would be -
      1 item found:
           ToString in num.ToString().ToUpper().PadLeft(5, ' ');
```

### Version Information

Supported from v1.8.1

## 5.130 CxList.GetMembersWithTargets Method (CxList, int)

Returns a CxList which is a subset of “this” instance with nodes that are part of a member/target pair (typical example: target.member) and have a direct target in "targets" parameter, or a target of target, a target of a target of a target …. Up to depthLimit depth

### Syntax

```
CxQL
public CxList GetMembersWithTargets (CxList targets, int depthLimit)
```

**Parameters**

targets - CxList of DOM objects which might be the target(s) of elements in "this", or the target of a target of this…
depthLimit – the number of iterations to look for targets

**Return Value**

Returns a CxList which is a subset of “this” instance with nodes that have a direct target.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Remarks

The return value may be empty (Count = 0).

If targets is null – returns an empty CxList

### Example

The following code example shows how you can use the GetMembersWithTargets method.

```
CxQL

 This example demonstrates the CxList. GetMembersWithTargets() method.
 The input source code is:

 int num = 55;
 string Str = num.ToString().ToUpper().PadLeft(5, ' ');

CxList methods = All.FindByType(typeof(MethodInvokeExpr));
CxList num = All.FindByShortName("num");
result = methods.GetMembersWithTargets(num, 2);
```

```
the result would be -
     2 items found:
          ToString, ToUpper in num.ToString().ToUpper().PadLeft(5, ' ');
```

### Version Information

Supported from v1.8.1

## 5.131 CxList.GetMethod Method (CxList)

Returns CxList which is a subset of this instance and its elements are methods of the specified CxList.

### Syntax

```
CxQL
public CxList GetMethod(CxList list)
```

**Parameters**

**List**

CxList of any DOM objects.

**Return Value**

A subset of this instance which contains methods of the specified CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the GetMethod method.

```
CxQL

 This example demonstrates the CxList. GetMethod() method.
 The input source code is:

 Class Cl
 {
       void foo()
       {
              int i = 1;
              i++;
       }
 }

CxList I_var = All.FindByShortName("i");
result = All. GetMethod(I_var);

 the result would be -
      1 item found:
             foo
```

### Version Information

Supported from v1.8.1

## 5.132 CxList.GetName Method ()

Returns a first data element name in requested CxList. Using to get internal data of first object in requested CxList

### Syntax

```
CxQL
public string GetName()
```

**Parameters**

none

**Return Value**

A name of the first element in Data. If CxList empty return null.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

None

### Example

```
CxQL

 This example demonstrates the CxList.GetName() method.
 The input source code is:

class cl1
 {
      void foo()
      {
             int a = 3;
             int b = 5;
      }
 }
result = All.FindByShortName("foo");
if (result.Count > 0)
       cxLog.WriteDebugMessage(result.GetName());

 the result would be on DebugMessage tab in CxAudit program
       foo
```

### Version Information

Supported from v1.8.1

## 5.133 CxList.GetParameters Method (CxList)

Returns a CxList which is a subset of this instance and its elements are parameters of methods elements provided in CxList.

### Syntax

```
CxQL
public CxList GetParameters (CxList MethodsList)
```

**Parameters**

**MethodList**

CxList of methods.

**Return Value**

Returns a CxList with all the parameters, from instance CxList, of the methods in MethodsLis.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

This example demonstrates the CxList.GetParameters(MethodsList) method.
The input source code is:

 foo(1, 3, i);

CxList methods = All.FindByType(typeof(MethodInvokeExpr));
result = All.GetParameters(methods);

 the result would consist of 3 items:
            1,
            3,
            i
```

### Version Information

Supported from  v1.8.1

## 5.134 CxList.GetParameters Method (CxList, int)

Returns a CxList which is a subset of instance CxList and its elements are parameters of methods elements provided in CxList.

### Syntax

```
CxQL
public CxList GetParameters (CxList MethodsList, int paramNo)
```

**Parameters**

**MethodList**

CxList of methods.

**paramNo**

The number of parameter to return (begins with 0)

**Return Value**

Returns a CxList with paramNo parameters, from instance CxList, of the methods in MethodsList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

This example demonstrates the CxList.GetParameters(MethodsList,paramNo) method.
The input source code is:

 foo(1, 3, i);

CxList methods = All.FindByType(typeof(MethodInvokeExpr));
result = All.GetParameters(methods, 1);

 the result would consist of 1 item:
            3
```

### Version Information

Supported from v1.8.1

## 5.135 CxList.GetParameters Method (CxList, int, ParameterIndexDirection)

Returns a CxList which is a subset of instance CxList and its elements are parameters of methods elements provided in CxList. The index integer is the position of the desired parameter. The direction refers to the index direction. If forward, 0 is the first parameter. If backwards, 0 is the last parameter.

### Syntax

```
CxQL
public CxList GetParameters (CxList MethodsList, int paramNo,
ParameterIndexDirection direction)
```

**Parameters**

**MethodList**

CxList of methods.

**paramNo**

The number of parameter to return (begins with 0)

**direction**

The direction for the parameter position. Can be `ParameterIndexDirection.Forward` or `ParameterIndexDirection.Backward`

**Return Value**

Returns a CxList with paramNo parameters, from instance CxList, of the methods in MethodsList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

This example demonstrates the CxList.GetParameters(MethodsList, paramNo,
direction) method. The input source code is:

 foo(1, 3, i);

CxList methods = All.FindByType<MethodInvokeExpr>();
result = All.GetParameters(methods, 0, CxList.ParameterIndexDirection.Backward);

 the result would consist of 1 item:
             i
```

### Version Information

Supported from v9.5.4

## 5.136 CxList.GetPathsOrigins Method ()

Returns a CxList which is a subset of instance CxList and contains end nodes of paths.

### Syntax

```
CxQL
public CxList GetPathsOrigins ()
```

**Return Value**

Returns CxList that contains end nodes of paths

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.GetPathsOrigins() method.
 The input source code is:

 public void setString (String str){
       if (str.length >0){
              lst.add(str);
       }
 }
CxList paths = All.DataInfluencingOn(All.FindByShortName("add"));
result = paths.GetPathsOrigins();

the result would consist of 3 items:
            lst      (in lst.add(str);)
              str    (in lst.add(str);)
              str    (in (String str);)
```

## 5.137 CxList.GetStartAndEndNodes Method (GetStartEndNodesType)

Returns CxList which is a subset of instance CxList and contains start nodes or end nodes or both start and nodes of path or all nodes in path.

### Syntax

```
CxQL
public CxList GetStartAndEndNodes (GetStartEndNodesType type)
```

**Parameters**

**Type**

The type of nodes to be returned:

CxList.GetStartEndNodesType.StartNodesOnly

CxList.GetStartEndNodesType.EndNodesOnly

CxList.GetStartEndNodesType.StartAndEndNodes

CxList.GetStartEndNodesType.AllNodes

**Return Value**

Returns  CxList which is a start nodes or end nodes or both start and nodes of path or all nodes in path.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | parameter is a null reference |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.GetStartsAndEndNodes (type) method.
 The input source code is:

public void setString (String str){
       lst.add(str);
}

CxList paths = All.DataInfluencingOn(All.FindByShortName("add"));

1. result =
paths.GetStartAndEndNodes(CxList.GetStartEndNodesType.StartNodesOnly);
the result would consist of 2 items:
            lst      (in lst.add(str);)
            str      (in (String str);)
2. result =
paths.GetStartsAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly);
the result would consist of 1 items:
              add    (in lst.add(str);)
3. result=
paths.GetStartAndEndNodes(CxList.GetStartEndNodesType.StartAndEndNodes);
the result would consist of 3 items:
            lst      (in lst.add(str);)
            str      (in (String str);)
              add    (in lst.add(str);)
4. result = paths.GetStartAndEndNodes(CxList.GetStartEndNodesType.AllNodes);
the result would consist of 4 items:
            lst      (in lst.add(str);)
            str      (in (String str);)
              add    (in lst.add(str);)
              str    (in lst.add(str);)
5. result =
paths.GetStartAndEndNodes(CxList.GetStartEndNodesType.AllButNotStartAndEnd);
the result would consist of 2 items:
            lst      (in lst.add(str);)
              str    (in lst.add(str);)
```

### Version Information

Supported from v7.1.2

## 5.138 CxList.GetFirstNodesInPath Method()

Returns CxList which is a subset of instance CxList and contains start nodes of path.

### Syntax

```
CxQL
public CxList GetFirstNodesInPath (GetStartEndNodesType type)
```

**Parameters**

none

**Return Value**

Returns CxList which is a start nodes of path.

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.GetFirstNodesInFlow () method.
 The input source code is:

public void setString (String str){
lst.add(str);
}

CxList paths = All.DataInfluencingOn(All.FindByShortName("add"));

// similar
to paths.GetStartAndEndNodes(CxList.GetStartEndNodesType.StartNodesOnly);
result = paths.GetFirstNodesInPath();
the result would consist of 2 items:
    lst(in lst.add(str);)
            str(in (String str);)
```

### Version Information

Supported from v9.4.0

## 5.139 CxList.GetLastNodesInPath Method ()

Returns CxList which is a subset of instance CxList and contains end nodes of path.

### Syntax

```
CxQL
public CxList GetLastNodesInPath (GetStartEndNodesType type)
```

**Parameters**

none

**Return Value**

Returns  CxList which is a end nodes of path.

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.GetLastNodesInFlow () method.
 The input source code is:

public void setString (String str){
lst.add(str);
}

CxList paths = All.DataInfluencingOn(All.FindByShortName("add"));

// similar
to paths.GetStartAndEndNodes(CxList.GetStartEndNodesType.EndNodesOnly);
result = paths.GetLastNodesInPath();
the result would consist of 1 items:
add(in lst.add(str);)
```

### Version Information

Supported from  v9.4.0

## 5.140 CxList.GetTargetOfMembers Method ()

Returns the list of elements which are the targets from the members of “this” instance.

### Syntax

```
CxQL
public CxList GetTargetOfMembers()
```

**Parameters**

none

**Return Value**

A list of objects from which “this” instance elements are member of.

### Example

```
CxQL
```

```
 This example demonstrates the CxList.GetTargetOfMembers() method.
 The input source code is:
 class cl1
 {
      void foo()
      {
             int a = obj.func();
      }
 }

result = All.FindByName("*.func").GetTargetOfMembers();

 The result would consist of 1 item:
      obj (in int a = obj.func())
```

## 5.141 CxList.GetTargetsWithMembers Method ()

Returns a CxList which is a subset of “this” instance with nodes that are part of a member/target pair (typical example: target.member) and have a direct member (i.e. they are the target).

### Syntax

```
CxQL
public CxList GetTargetsWithMembers()
```

**Return Value**

Returns a CxList which is a subset of “this” instance with nodes that have a direct member.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Remarks

The return value may be empty (Count = 0).

### Example

The following code example shows how you can use the GetTargetsWithMembers method.

```
CxQL

 This example demonstrates the CxList.GetTargetsWithMembers() method.
 The input source code is:

 int num = 55;
 string Str = num.ToString().ToUpper().PadLeft(5, ' ');

CxList methods = All.FindByType(typeof(MethodInvokeExpr));
result = methods.GetTargetsWithMembers();

 the result would be -
      2 items found:
            ToUpper, ToString in num.ToString().ToUpper().PadLeft(5, ' ');
```

### Version Information

Supported from v1.8.1

## 5.142 CxList.GetTargetsWithMembers Method (CxList)

Returns a CxList which is a subset of “this” instance with nodes that are part of a member/target pair (typical example: target.member) and have a direct member in the CxList paramemeter "members".

### Syntax

```
CxQL
public CxList GetTargetsWithMembers(CxList members)
```

**Parameters**

members - CxList of DOM objects which might be the member(s) of elements in "this"

**Return Value**

Returns a CxList which is a subset of “this” instance with nodes that have a direct member in members parameter.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Remarks

The return value may be empty (Count = 0).

If members is null – returns an empty CxList

### Example

The following code example shows how you can use the GetTargetsWithMembers method.

```
CxQL

 This example demonstrates the CxList.GetTargetsWithMembers() method.
 The input source code is:

 int num = 55;
 string Str = num.ToString().ToUpper().PadLeft(5, ' ');

CxList methods = All.FindByType(typeof(MethodInvokeExpr));
CxList member = All.FindByShortName("PadLeft");
result = methods.GetTargetsWithMembers(member);

 the result would be -
      1 item found:
            ToUpper in num.ToString().ToUpper().PadLeft(5, ' ');
```

### Version Information

Supported from v1.8.1

## 5.143 CxList.GetTargetsWithMembers Method (CxList, int)

Returns a CxList which is a subset of “this” instance with nodes that are part of a member/target pair (typical example: target.member), and have a direct member in CxList paramemeter "members", or a member of a member… up to depth depthLimit

### Syntax

```
CxQL
public CxList GetTargetsWithMembers(CxList targets, int depthLimit)
```

**Parameters**

members - CxList of DOM objects which might be the member(s) of elements in "this",
or the memebr of member of this, …

depthLimit – the number of iterations to look for members

**Return Value**

Returns a CxList which is a subset of "this" instance with nodes that have a direct /chain memebr.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Remarks

The return value may be empty (Count = 0).

If members is null – returns an empty CxList

### Example

The following code example shows how you can use the GetTargetsWithMembers method.

```
CxQL

 This example demonstrates the CxList.GetTargetsWithMembers() method.
 The input source code is:

string Str = "sample".ToString().ToUpper().PadLeft(5, ' ');

CxList methods = All.FindByType(typeof(MethodInvokeExpr));
CxList member = All.FindByShortName("PadLeft");
result = methods.GetTargetsWithMembers(member, 2);

 the result would be -
      2 items found:
           ToString, ToUpper in num.ToString().ToUpper().PadLeft(5, ' ');
```

### Version Information

Supported from  v1.8.1

## 5.144 CxList.InheritsFrom Method (string)

Returns a CxList which is a subset of "this" instance and its elements are inherited from the given class name.

### Syntax

```
CxQL
public CxList InheritsFrom(string baseClassName)
```

**Parameters**

**baseClassName**

The name of the base class.

**Return Value**

A subset of "this" instance which elements are inherited from the given base class name.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.InheritsFrom() method.
 The input source code is:

 class BClass
 {

 }

 class CClass : BClass
 {

 }

result = All.InheritsFrom("BClass");

 The result would consist of 1 item:
            CClass
```

### Version Information

Supported from  v1.8.1

## 5.145 CxList.InheritsFrom Method (CxList)

Returns a CxList which is a subset of "this" instance and its elements are inherited from the given CxList of classes.

### Syntax

```
CxQL
public CxList InheritsFrom(CxList baseClassList)
```

**Parameters**

**baseClassList**

The CxList of base classes.

**Return Value**

A subset of "this" instance which elements are inherited from the given base classes.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.InheritsFrom() method.
 The input source code is:

 class BClass
 {

 }

 class CClass : BClass
 {

 }

CxList cl = All.FindByName("BClass");
result = All.InheritsFrom(cl);

 The result would consist of 1 item:
            CClass
```

### Version Information

Supported from v1.8.1

## 5.146 CxList.InheritsFrom Method (params string[])

Returns a CxList which is a subset of this instance and its elements are inherits from the specified List of classes.

### Syntax

```
CxQL
public CxList InheritsFrom(params string[] baseClassNames, bool inheritsFromAll=
false)
```

**Parameters**

**baseClassNames**

List of names of the base classes.

**inheritsFromAll**

By default is **false**. The instance should inherit from all list baseClassNames, works at the first level of inheritance.

**Return Value**

A subset of “this” instance which elements are inherited from the given base class name.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL
 This example demonstrates the CxList.InheritsFrom() method.
 The input source code is:

 class BClass
 {

 }

 class AClass
 {

 }

 class CClass : BClass
 {

 }

 class DClass : BClass, AClass
 {

 }


result = All.InheritsFrom(new List<string>{"BClass"});

 The result would consist of 2 itens:
    CClass, DClass

result = All.InheritsFrom(new List<string>{"BClass", “AClass”});

 The result would consist of 1 item:
    DClass
```

### Version Information

Supported from v9.4

## 5.147 CxList.FindDescendantsOfType<T> Method (CxList)

Returns a CxList which contains the direct descendants of a given instance members, filter by type.

### Syntax

```
CxQL
public CxList FindDescendantsOfType<T>(CxList ancs)
```

**Parameters**

<T>

the type to cast the DOM object to (must inherit from CSharpGraph

**ancs**

The Ancestors whose descendants are to be returned.

**Return Value**

Return a CxList with the direct descendants of a given instance members, filter by type.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL
 This example demonstrates the CxList.InheritsFrom() method.
 The input source code is:

 class BClass
 {
Int field1;
Float field2;
 }

classList  = Find_ClassDecl();
result = All.FindDescendants<FieldDecl>(classList);

 The result would consist of 2 itens:
    field1, field2
```

### Version Information

Supported from v9.4

## 5.148 CxList.FindDescendantsOfType Method (CxList, Type)

Returns a CxList which contains the direct descendants of a given instance members, filter by type.

### Syntax

```
CxQL
public CxList FindDescendantsOfType<T>(CxList ancs, Type type)
```

**Parameters**

**ancs**

The Ancestors whose descendants are to be returned.

**type**

the type of the DOM object <

**Return Value**

Return a CxList with the direct descendants of a given instance members, filter by type.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Comments

The return value may be empty (Count = 0).

### Example

```
CxQL
 This example demonstrates the CxList.InheritsFrom() method.
 The input source code is:

 class BClass
 {
Int field1;
Float field2;
 }

classList  = Find_ClassDecl();
result = All.FindDescendants(classList, typeof(FieldDecl));

 The result would consist of 2 itens:
    field1, field2
```

### Version Information

Supported from  v9.4

## 5.149 CxList.IntersectWithNodes Method (CxList)

Returns a CxList which is a subset of paths, which are the instance CxList, that includes elements of intersected CxList.

### Syntax

```
CxQL
public CxList IntersectWithNodes (CxList intersect)
```

**Parameters**

`intersect`

intersected CxList elements

**Return Value**

Returns a CxList  which is a subset of the 'this' instance , that includes at least one element of intersected CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL
```

```
 This example demonstrates the CxList.IntersectWithNodes() method.
 The input source code is:

 public void setString (String str){
       if (str.length >0){
              lst.add(str);
       }
       else{
              String otherStr ="string is empty";
              lst.add(otherStr);
       }
}

CxList intersect = All.FindByShortName("otherStr");
CxList paths = All.DataInfluencingOn(All.FindByShortName("add"));
result = paths.IntersectWithNodes(intersect);
the result would consist of 3 items:
       all ending at add    (in lst.add(otherStr);)
       starting
            otherStr        (in lst.add(otherStr);)
              otherStr             (in String otherStr ="string is empty";)
              "string is empty"    (in String otherStr ="string is empty";)
```

### Version Information

Supported from 7.1.2

## 5.150 CxList.IntersectWithNodes Method (CxList, CxList.IntersectionType)

Returns a CxList which is a subset of paths, which are the instance CxList, that includes elements of intersected CxList.

### Syntax

CxQL

```
public CxList IntersectWithNodes (CxList intersect, IntersectionType type)
```

**Parameters**

**`intersect`**

intersected CxList elements

**`type`**

The type of intersection to be made:

CxList.IntersectionType.AllNodes

CxList.IntersectionType.AnyNodes (default)

**Return Value**

Returns a CxList which is a subset of the ‘this’ instance , that includes at least one element of intersected CxList when the type is ‘AnyNodes’, and includes all elements of intersected CxList when the type is ‘Allnodes’.

### Exceptions

| Exception type | Condition |
|---|---|

| | |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList.IntersectWithNodes() method.
 The input source code is:

 public void setString (String str){
       if (str.length >0){
              lst.add(str);
       }
       else{
              String otherStr ="string is empty";
              lst.add(otherStr);
       }
}

CxList intersect = All.FindByShortName("otherStr");
intersect.Add(All.FindByType(typeof(StringLiteral)));
CxList paths = All.DataInfluencingOn(All.FindByShortName("lst"));
result = paths.IntersectWithNodes(intersect, CxList.IntersectionType.AllNodes);
the result would consist of 1 item:
       a flow starting on “string is empty”, passing on the 2 occurrences of
‘otherStr’ and the ‘add’ method, and ending on the ‘lst’ UnknownReference.
```

### Version Information

Supported from 9.1.0

## 5.151 CxList.ReduceFlow Method (CxList.ReduceFlowType)

Returns CxList which is a subset of instance CxList and consists of longest paths to/from destination element for CxList.ReduceFlowType.ReduceSmallFlow parameter or shortest paths to/from destination element for CxList.ReduceFlowType.ReduceBigFlow parameter.

### Syntax

```
CxQL
public CxList ReduceFlow (CxList.ReduceFlowType flowType)
```

**Parameters**

**Type**

The type of flow for reduce:

CxList.ReduceFlowType.ReduceBigFlow

CxList.ReduceFlowType.ReduceSmallFlow

**Return Value**

Returns CxList which is a subset of paths that consists of longest paths or shortest paths to/from destination element, depending on ReduceFlow methods parameter.

## Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

## Remarks

The return value may be empty (Count = 0).

## Example

```
CxQL

 This example demonstrates the CxList. ReduceFlow () method.

 The input source code is:

ArrayList<String> lst = new ArrayList<String>();
public void setString (String str){
       if (str.length >0){
              lst.add(str);
       }
       else{
              String otherStr ="string is empty";
              lst.add(otherStr);
       }
}

CxList paths = All.DataInfluencingOn(All.FindByShortName("add"));

1.result =  paths.ReduceFlow(CxList.ReduceFlowType.ReduceBigFlow);
the result would consist of 4 items:
       all ending at add    (in lst.add(otherStr);)
       starting
              lst                  (in lst.add(str))
              str                  (in lst.add(str))
              lst                  (in lst.add(otherStr);)
            otherStr        (in lst.add(otherStr);)

2. result = paths.ReduceFlow(CxList.ReduceFlowType.ReduceSmallFlow);

the result would consist of 4 items:
       all ending at add    (in lst.add(otherStr);)
starting      lst    (in ArrayList<String> lst = new ArrayList<String>();)
ending add           (in lst.add(str);)

starting      lst           (in lst.add(str))
ending add           (in lst.add(otherStr);)

starting      str           (in (String str))
ending add           (in lst.add(str);)

starting      "string is empty"           (in String otherStr ="string is
empty";)
ending add                         (in lst.add(otherStr);)
```

## Version Information

Supported from 7.1.2

## 5.152 CxList.ReduceFlowByPragma Method ()

Returns a CxList which is a subset of instance CxList and consists of shortest paths from path starting line to path end line.

### Syntax

```
CxQL
public CxList ReduceFlowByPragma ()
```

**Parameters**

**Return Value**

Returns a CxList which are shortest paths from path starting line to path end line.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList. ReduceFlowByPragma () method.
 The input source code is:

 public void setString (){
       String otherStr = otherStr;
       lst.add(otherStr);
}

CxList paths = All.DataInfluencedBy(All.FindByShortName("otherStr"));
result = paths.ReduceFlowByPragma();

the result would consist of 4 items:
  starts in otherStr of (String otherStr) ends in otherStr of
(lst.add(otherStr);)
  starts in otherStr of (= otherStr;)     ends in otherStr of (String otherStr)
  starts in otherStr of (String otherStr) ends in add of (lst.add(otherStr);)
  starts in otherStr of (lst.add(otherStr);) ends in add of (lst.add(otherStr);)
```

### Version Information

Supported from 7.1.2

## 5.153 CxList.SanitizeCxList Method (CxList sanitizeNodes)

Returns a CxList which is a subset of paths, which are the instance CxList, that doesn't include sanitize nodes.

### Syntax

```
CxQL
public CxList SanitizeCxList (CxList sanitizeNodes)
```

**Parameters**

**SanitizeNodes**

CxList of sanitizer nodes.

**Return Value**

Returns a CxList which is a subset of paths that doesn't include sanitize nodes.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

The return value may be empty (Count = 0).

### Example

```
CxQL

 This example demonstrates the CxList. SanitizeCxList () method.
 The input source code is:

public void setString (String input){
       String otherStr = input;
       lst.add(otherStr);
}
CxList paths = All.DataInfluencingOn(All.FindByShortName("add"));
CxList sanitizeNodes = All.FindByShortName("input");
result = paths.SanitizeCxList(sanitizeNodes);

the result would consist of 3 items:
       all ends with add in lst.add(otherStr);
       starts:
       otherStr      (in String otherStr = input;)
       lst           (in lst.add(otherStr);)
       otherStr      (in lst.add(otherStr);)
```

### Version Information

Supported from 7.1.2

## 5.154 CxList.FillGraphsList Method (CxList)

Fills graphs for the list of roots given.

### Syntax

```
CxQL
public void FillGraphsList (CxList graphRoots)
```

**Parameters**

**graphRoots**

List of roots to be filled with the graphs.

**Return Value**

None.

### Exceptions

| Exception type | Condition |
|---|---|

| | |
|---|---|
| NullReferenceException | **parameter is a null reference** |

### Example

```
CxQL

 This example demonstrates the CxList. FillGraphsList () method.
 With any Input source code, the method can be called after a Query.
result=All;
FillGraphsList(result);
 At this point, the result list is filled with the Graphs.
```

### Version Information

Supported from 7.1.2

## 5.155 CxList.FillGraphsList Method (CSharpGraph)

Fill graphs from one root element.

### Syntax

```
CxQL
public void FillGraphsList (CSharpGraph graphRoot)
```

**Parameters**

**graphRoot**

CSharpGraph instance to be filled with Graphs.

**Return Value**

None.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Example

```
CxQL

 This example demonstrates the CxList. FillGraphsList () method.
 With any Input source code, the method can be called after a Query.
first=All.GetFirstGraph();
FillGraphsList(first);
 At this point, the first is filled with the Graphs.
```

## 5.156 CxList.GetIndexOfParameter Method ()

For a single Param or ParamDecl returns the index of the parameter 0 based.

### Syntax

```
CxQL
public int GetIndexOfParameter ()
```

**Parameters**

**Return Value**

Integer containing the index of the parameter zero based, or -1 if not a parameter or list empty or contains multiple nodes. Note that the CxList must contain exactly one node, and the node must be of type Param or ParamDecl.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

 This example demonstrates the CxList.GetIndexOfParameterMethod() method.
 It prints out to the log the index of each of the parameters.

result = All.FindByType(typeof(Param));
foreach (CxList list in result)
{
       cxLog.WriteDebugMessage("Parameter index = " + list.
GetIndexOfParameter());
}
```

## 5.157 CxList.FindSQLInjections Method (CxList, CxList, CxList)

Returns flow for SQL Injection from input to db that is not sanitized

### Syntax

```
CxQL
public CxList FindSQLInjections (CxList inputs, CxList db, CxList sanitize)
```

**Parameters**

**inputs**

CxList containing input elements

**db**

CxList containing output elements (eg. database)

**sanitize**

CxList containing sanitizing elements (cast to integer etc).

**Return Value**

CxList containing flow of SQL injection from input to output which is not flowing through a sanitizer

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Remarks

Actually uses inputs.InfluencingOnAndNotSanitized(db, sanitize).

### Example

```
CxQL
```

```
// Find the inputs for the SQL injection
CxList inputs = All.FindByShortName("ReadLine");
//Find the entrance to the Database command
CxList dbIn = All.GetParameters(All.FindByShortName("SqlCommand"));
//Find a potential sanitizer
CxList integerSanitizers = Find_Integers();
//Find sql injections using previous results
result = All.FindSQLInjections(inputs, dbIn, integerSanitizers);
```

## 5.158 CxList.FindXSS Method (CxList, CxList, CxList)

Returns flow for XSS from input to output that is not sanitized

### Syntax

```
CxQL
public CxList FindXSS (CxList inputs, CxList outputs, CxList sanitize)
```

**Parameters**

**inputs**

CxList containing input elements

**outputs**

CxList containing output elements for xss.

**sanitize**

CxList containing sanitizing elements (cast to integer etc).

**Return Value**

CxList containing flow of XSS from input to output which is not flowing through a sanitizer

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Remarks

Actually uses inputs.InfluencingOnAndNotSanitized(db, sanitize).

### Example

```
CxQL

CxList inputs = All.FindByShortName("request");
CxList outputs = All.FindByShortName("output");
CxList sanitize = All.FindByShortName("escape");
result = All.FindXSS(inputs, outputs, sanitize);
```

## 5.159 CxList.Clone Method ()

Clones the current (this) CxList

### Syntax

```
CxQL
public CxList Clone ()
```

**Parameters**

**Return Value**

CxList containing a clone of the current (this) CxList

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

CxList A = All.FindByType(typeof(UnknownReference));
CxList B = A; //B points to same elements as A
CxList B = A.Clone(); //B has a copy (clone) of the elements in A
```

## 5.160 CxList.TryGetCSharpGraph<T> Method () where T : CSharpGraph

Try to extract the DOM object from the first node in 'this' CxList and cast it to type 'T'. Returns null if the CxList is empty, or if the casting fails.

### Syntax

```
CxQL
public CxList TryGetCSharpGraph <T>()
```

**Parameters**

<T> the type to cast the DOM object to (must inherit from CSharpGraph)

**Return Value**

The DOM object after casting

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

CxList A = All.FindById(10);
CSharpGraph cs = A.TryGetCSharpGraph<CSharpGraph>();
// If A contains at least 1 node, cs will contain its DOM object
```

### Version Information

Supported from 8.4.0

## 5.161 CxList.GetQueryParam Method (string paramName)

Try to get a value for a query parameter using the key paramName. Returns an empty string if the key was not found and on errors.

### Syntax

```
CxQL
```

```
public string GetQueryParam(string paramName)
```

**Parameters**

paramName

The parameter name (key)

**Return Value**

The value of the received key, or an empty string if the key was not found.

## Exceptions

| Exception type | Condition |
|---|---|
| | |

## Example

```
CxQL

string val = All.GetQueryParam("Param");
// If the key "Param" was found in the configuration, than val now holds its
value, otherwise, val is an empty string
if (string.IsNullOrEmpty(val))
{
	// Use val
}
```

## Version Information

Supported from 8.4.0

## 5.162 CxList.GetQueryParam<T> Method (string paramName, T defaultVal = default(T) )

Try to get a value for a query parameter using the key paramName and parse the returnd string value to type T. Returns defaultVal if the key was not found and on errors.

## Syntax

```
CxQL
public string GetQueryParam<T>(string paramName, T defaultVal = default(T))
```

**Parameters**

<T>

The type of defaultVal and the returned value

paramName

The parameter name (key)

defaultVal

The value to return on errors

**Return Value**

The value for the received key parset to type T, or defaultVal if the key was not found or if the value returned cannot be parsed to type T..

## Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

int val = All.GetQueryParam<int>("Param", 0);
// If the key "Param" was found in the configuration, than val now holds its
value, otherwise, val is 0
if (val > 0)
{
       // Use val
}
```

### Version Information

Supported from 8.4.0

## 5.163 CxList.FindByFiles Method (CxList source)

Return a subset of 'this' instance, where its DOM objects are on the same file(s) as the DOM objects in the 'source' CxList.

### Syntax

```
CxQL
public CxList FindByFiles(CxList source)
```

**Parameters**

source

A Cxlist that have DOM objects in the required files.

**Return Value**

Rerurns Return a subset of 'this' instance, where its DOM objects are on the same file(s) as the DOM objects in the 'source' CxList..

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

CxList a = All.FindByFileName("*method.js*");
CxList b = All.FindByFileName("*method.json");
Result = a.FindByFiles(b);
// Return a subset of 'a' where the objects are of the same file as the objects
of b.
```

### Version Information

Supported from 8.4.0

## 5.164 CxList.FindByFileNames Method (params string[])

Returns a CxList, which is a subset of “this” instance and its elements are in a given array of source code files.

## Syntax

```
CxQL
public CxList FindByFileNames(params string[] FileNames)
```

**Parameters**

**FileNames**

Parameters with file names.

**Return Value**

A subset of "this" instance with elements from a given list of file names.

## Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

## Comments

The return value may be empty (Count = 0).

## Example

```
CxQL
 This example demonstrates the CxList.FindByFileNames() method.
 The input source code is:

 //file myCode1.cs
 class Cl1 {
void foo() {
int i;
}
 }

//file myCode2.cs
 class Cl2 {
void bar() {
int i;
}
 }

result = All.FindByFileNames("*myCode1.cs", "*myCode2.cs");

 the result consists of 10 items:
     Cl1
     void,
     foo,
      int,
      i,
             Cl2
     void,
     bar,
      int,
      i
```

## Version Information

Supported from v9.4

## 5.165 CxList.FindRegexMatches Method (CxList comments)

Return a subset of 'this' instance which objects are of type Comment, and are equivalent to objects of type Comment in the comments CxList.

### Syntax

```
CxQL
public CxList FindRegexMatches(CxList comments)
```

**Parameters**

A CxList of Comment type objects to find matches against the Comment objects in 'this'

**Return Value**

Return a subset of 'this' instance which objects are of type Comment, and are equivalent to objects of type Comment in the comments CxList..

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

// Each time a FindByRegexExt or FindByRegex generate Comment type objects, they
are get a different NodeId, even if the represent the same string in the project
code.

CxList a = All.FindByRegexExt("http://");
a.Add(All.FindByRegexExt("https://"));
CxList b = All.FindByRegexExt("http://"); // The strings that starts with
http:// in 'a', now exist in 'b' but with a different NodeId number.
Result = a. FindRegexMatches(b);
// Return a subset of 'b' where the objects returned are equivalent to other
objects in 'a'.
```

### Version Information

Supported from 8.4.0

## 5.166 CxList.GetAssigner Method (CxList others = null)

For each DOM object in 'this' which is on the left side of an assignment, return the right side of the assignment, which are in the others CxList.

### Syntax

```
CxQL
public CxList GetAssigner(CxList others = null)
```

**Parameters**

others

CxList containing the right side of the assignment. If null – treat it as if it was All.

**Return Value**

For each DOM object in 'this' which is on the left side of an assignment, return the right side of the assignment, which are in the others CxList

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

The input source code is:
 int a = 0;
 int b = a;
 b = 2;
 int c = b > 1 ? 3 : a;
CxList a = All.FindByShortName("a");
CxList b = All.FindByShortName("b");
CxList c = All.FindByShortName("c");

result = b.GetAssigner();
// result now holds 'a' in int b = a; and 2 in b = 2;
result = c.GetAssigner();
// result now holds 3 and 'a' in int c = b > 1 ? 3 : a;
result = c.GetAssigner(a);
// result now holds 'a' in int c = b > 1 ? 3 : a;
```

### Version Information

Supported from 8.4.0

## 5.167 CxList.GetAssignee Method (CxList others = null)

For each DOM object in 'this' which is on the right side of an assignment, return the left side of the assignment, which are in the others CxList.

### Syntax

```
CxQL
public CxList GetAssignee(CxList others = null)
```

**Parameters**

others

CxList containing the left side of the assignment. If null – treat it as if it was All.

**Return Value**

For each DOM object in 'this' which is on the right side of an assignment, return the left side of the assignment, which are in the others CxList

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

The input source code is:
 int a = 0;
 int b = a;
 b = 2;
 int c = b > 1 ? 3 : a;
```

```
CxList a = All.FindByShortName("a");
CxList b = All.FindByShortName("b");
CxList c = All.FindByShortName("c");

result = a.GetAssignee();
// result now holds 'b' in int b = a; and c in int c = b > 1 ? 3 : a;
result = a.GetAssignee(b);
// result now holds 'b' in int b = a;
```

### Version Information

Supported from 8.4.0

## 5.168 CxList.GetAssignee Method (int assigneeNo)

For each DOM object in 'this' which is on the right side of an assignment, return the corresponding node that is the assigneeNo given on the left side of the assignment.

### Syntax

```
CxQL
public CxList GetAssignee(int assigneeNo)
```

**Parameters**

assigneeNo

int corresponding to the index of the assignees on the left – if -1 treat as All

**Return Value**

For each DOM object in 'this' which is on the right side of an assignment, return the corresponding node that is the assigneeNo given on the left side of the assignment.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

The input source code is:
 a,b := something();
CxList a = All.FindByShortName("something");

result = a.GetAssignee(0);
// result now holds 'a' in a,b := something();
result = a.GetAssignee(1);
// result now holds 'b' in a,b := something();
result = a.GetAssignee(-1);
// result now holds ‘a’ and 'b' in a,b := something();
result = a.GetAssignee(2);
// result now is empty in a,b := something();
```

### Version Information

Supported from 9.4.0

## 5.169 CxList.GetAssignee Method (CxList others = null, int assigneeNo = -1)

For each DOM object in 'this' which is on the right side of an assignment, return the left side of the assignment, which are in the others CxList, or the corresponding node that is the assigneeNo given on the left side of the assignment.

### Syntax

```
CxQL
public CxList GetAssignee(CxList others = null, int assigneeNo = -1)
```

**Parameters**

others

CxList containing the left side of the assignment. If null – treat it as if it was All.

assigneeNo

int corresponding to the index of the assignees on the left – if -1 treat as All

**Return Value**

For each DOM object in 'this' which is on the right side of an assignment, return the left side of the assignment, which are in the others CxList, or the corresponding node that is the assigneeNo given on the left side of the assignment.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

The input source code is:
 int a = 0;
 int b = a;
 b = 2;
 int c = b > 1 ? 3 : a;
 d,e := something();
CxList a = All.FindByShortName("a");
CxList b = All.FindByShortName("b");
CxList c = All.FindByShortName("c");

result = a.GetAssignee();
// result now holds 'b' in int b = a; and c in int c = b > 1 ? 3 : a;
result = a.GetAssignee(b);
// result now holds 'b' in int b = a;

CxList d = All.FindByShortName("something");

result = d.GetAssignee(0);
// result now holds 'd' in d,e := something();
result = d.GetAssignee(1);
// result now holds 'b' in d,e := something();
result = d.GetAssignee(-1);
```

```
// result now holds ‘d’ and 'e' in d,e := something();
result = d.GetAssignee(2);
// result now is empty in d,e := something();
```

### Version Information

Supported from 9.4.0

## 5.170 CxList Abstract Interpretation Methods

An abstract value is an instance of a class that implements the “IAbstractValue” interface.

The interface “IAbstractValue” is implemented by the following classes:

- AnyAbstractValue
- IntegerIntervalAbstractValue
- StringAbstractValue
- TrueAbstractValue
- FalseAbstractValue
- NullAbstractValue
- ObjectAbstractValue
- FunctionAbstractValue
- NoneAbstractValue

The ObjectAbstractValue class has two parameters which can be used to query a certain object:

- **ParentPointerMayBeNull**: indicated that the pointer's / instance's possible values contain the null value)
- **AllocationState**: current allocation state in the heap, value may be one the following:
  - ObjectAllocationState.Allocated (object is allocated and exists in the heap)
  - ObjectAllocationState.Released (object has been freed / released from the heap)
  - ObjectAllocationState.Ambiguous (object either exists or has been released from the heap)

bool IAbstractValue.IncludedIn(IAbstractValue absValue, bool strictTypeMatch = false)

**Parameters**

**abstractValue**

An abstract value

**strictTypeMatch**

**Return Value**

If “this” includes the abstract value parameter “absValue” → “IncludedIn” method returns true, otherwise method returns false.

Examples:

Here is a list of IAbstractvalue

```
List<IAbstractValue> list = {new AnyAbstractValue(),
                             new IntegerIntervalAbstractValue(1, 2),
                             new StringAbstractValue("body")};
```

1.
   ```
   int count = 0;
   IAbstractvalue source = new IntegerAbstractvalue(1);
   ```

```
foreach (var val in list)
        if (source.IncludedIn(val))  count++;
```

The value of variable count will be 2. // any and integer return “true”

2.
```
int count = 0;
IAbstractvalue source = new IntegerAbstractvalue(1);
foreach (var val in list)
        if (source.IncludedIn(val,true)) count++;
```

The value of variable count will be 1. // only integer return “true”

3.
```
int count = 0;
IAbstractvalue source = new IntegerAbstractvalue(1);
foreach (var val in list)
        if (val.IncludedIn(source))  count++;
```

The value of variable count will be 0. // [1,2] not included in [1,1]. Here is the question (IncludeIn) activate with opposite order.

### 5.170.1 CxList.FindByAbstractValue Method (Func<IAbstractValue, bool> criterion)

Returns a CxList which is a subset of this instance whose elements have an abstract value that fulfills the criterion.

#### Syntax

CxQL
```
public CxList FindByAbstractValue (Func<IAbstractValue, bool> criterion)
```

**Parameters**

criterion

Lambda method that can filter required items from this CxList according to their abstract value. This function have one parameter of type IAstractValue and returns bool.

**Return Value**

A subset of this instance whose elements have match the requested creterion.

#### Exceptions

| Exception type | Condition |
|---|---|
| | |

#### Example 1

CxQL

_Find all DOM elements whose abstract value is an integer inside the interval [0,10]_
```
IAbstractValue intervalZeroToTen = new IntegerIntervalAbstractValue(0,10);
CxList res = All.FindByAbstractValue(abstractValue =>
                          abstractValue.IncludedIn(intervalZeroToTen,true));
```

_Find all DOM elements for which the integer 0 in inside their abstract value_
```
IAbstractValue zero = new IntegerIntervalAbstractValue(0);
CxList res = All.FindByAbstractValue(abstractValue =>
                                     zero.IncludedIn(abstractValue));
```

_Find all DOM elements whose abstract value has a given type_

```
CxList res = All.FindByAbstractValue(abstractValue =>
                                     abstractValue is StringAbstractValue);
```

### Example 2

```
CxQL

The input source code is:
 int counter = 0;
 int x = counter + 5;
 string str = "a";
 string secondStr = str + "b";
CxList a = All. FindByAbstractValue(abstractValue =>
                 abstractValue is IntegerIntervalAbstractValue);
CxList b = All. FindByAbstractValue (abstractValue =>
                 abstractValue is StringAbstractValue);

result = a;
// result now holds 0, '+', 'counter' and 5  in
 int counter = 0;
 int x = counter + 5;
result = b;
// result now holds "a", '+', 'str' and "b" in
 string str = "a";
 string secondStr = str + "b";
```

### Example 3

```
CxQL

The input source code is:
 var y;
 int counter = 0;
 int x = counter + 5;
 y();
IAbstractValue zeroAbsValue = new IntegerIntervalAbstractValue(0);
result = All. FindByAbstractValue(abstractValue =>
                 zeroAbsValue.IncludedIn(abstractValue));
/* result now holds 0 and 'counter' in
   int x = counter + 5;
   And also contains y in: (because y has AnyAbstractValue which includes 0)
   y();
*/

result = All. FindIncludedAbstractValue(abstractValue =>
                 zeroAbsValue.IncludedIn(abstractValue,true));
// result now holds 0 and 'counter' in
 int x = counter + 5;
// it does not contain y because we asked for strictTypeMatch
```

### Example 4

```
CxQL

The input source code is:
 int counter = 0;
 int x = counter + 5;
 function foo() {} // void method
 foo();
IAbstractValue zeroToFiveAbsValue = new IntegerIntervalAbstractValue(0,5);
result = All. FindIncludedInAbstractValue(abstractValue =>
                 abstractValue.IncludedIn(zeroToFiveAbsValue));
```

```
/* result now holds 0, 5, 'counter' and the sum (counter + 5) in
   int counter = 0;
   int x = counter + 5;
   And also contains foo in:
   foo (); // because y has AnyAbstractValue which includes [0,5]
*/

result = All. FindIncludedInAbstractValue(abstractValue =>
                    abstractValue.IncludedIn(zeroToFiveAbsValue, true));
// result now holds 0, 5, 'counter' and the sum of (counter + 5) in
   int x = counter + 5;
// it does not contain foo because we asked for strictTypeMatch
```

### Version Information

Supported from 8.6.0

## 5.170.2 CxList.FindByAbstractValues Method (CxList sources)

Returns a CxList which is a subset of this instance whose elements have an abstract value equal to the abstract value of one element in the sources CxList.

### Syntax

```
CxQL
public CxList FindByAbstractValues(CxList sources)
```

**Parameters**

sources

A CxList.

**Return Value**

A subset of this instance whose elements have an abstract value equal to the abstract value of one element in the sources CxList.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

The input source code is:
 string str = "a";
 string secondStr = str + "b";
result = All.FindByAbstractValues(All.FindByType(typeof(StringLiteral)));
// result now holds "a", 'str' and "b" in
 string str = "a";
 string secondStr = str + "b";
```

### Version Information

Supported from 8.4.2

## 5.171 Scan Provider Methods

### 5.171.1 cxScan.IsFrameworkActive Method (string frameworkName)

Returns bool if requested framework present in scaned project..

#### Syntax

```
CxQL
public bool IsFrameworkActive (string frameworkName)
```

**Parameters**

string

Name of requested framework

**Return Value**

True → requested framework present in this project.

False → othewise

#### Exceptions

| Exception type | Condition |
|---|---|
| | |

#### Example

```
CxQL

Implementation of query JavaScript Kony_Code_Injection

if(cxScan.IsFrameworkActive("Kony"))
{
        CxList inputs = Kony_UI_Inputs();
        CxList Eval = Find_Outputs_CodeInjection();
        CxList sanitize = Code_Injection_Sanitize();
        and etc.
}

if “Kony” framework present →this query do something, otherwise do nothing
You can test existing of every framework.
```

#### Version Information

Supported from 8.6.0

### 5.171.2 cxScan.GetScanProperty Method (string key)

Returns string with a value of requested property..

#### Syntax

```
CxQL
public string GetScanProperty (string key)
```

**Parameters**

string

Name of requested property, currently only `projectPath` property supported.

**Return Value**

If requested property exists → value of requested property will be returned.

Otherwise empty string will be returned.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL


string projectPath = cxScan.GetScanProperty("projectPath");

string projectPath contains value of "projectPath" property.
```

### Version Information

Supported from 8.6.0

## 5.172 CxList CxXPath Methods

### 5.172.1 IEnumerable<CxXmlDoc> GetXmlFiles Method (string filter, bool IgnoreNamespaces = false)

Returns an enumerated list of XML files filtered according to the first parameter.

### Syntax

```
CxQL
public IEnumerable<CxXmlDoc> GetXmlFiles (string filter, bool IgnoreNamespaces =
false)
```

**Parameters**

string

File extension pattern to be used as filter.

bool

Specifies whether the search should ignore the namespaces. This field is not required and its default value is false.

**Return Value**

Returns an enumerated list of XML files.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

IEnumerable xmlDoc = cxXpath.GetXmlFiles("*.cx", true);

Returns an enumerated list of XML files filtered by "*.cx"
```

### Version Information

Supported from 8.6.0

### 5.172.2 CxList CreateXmlNode Method (XPathNavigator input, CxXmlDoc xmlDoc, int language, bool save, int depth = 1)

Return a CxList composed by CxXmlNode.

#### Syntax

```
CxQL
public CxList CreateXmlNodes(XPathNavigator input, CxXmlDoc xmlDoc, int language,
bool save, int depth = 1)
```

**Parameters**

XPathNavigator

Provides a cursor model for navigating XML data.

CxXmlDoc

Document where the node will be created.

int

Id of the language.

bool

Sprecifies if the node should be saved.

int

Deapth of search. Default value is 1.

**Return Value**

Return a CxList with CxXmlNodes for the given XPath.

#### Exceptions

| Exception type | Condition |
|---|---|
| | |

#### Remarks

1. Save option set to true is deprecated since 9.4.0.

#### Example

```
CxQL

// create an XPathDocument object
XPathDocument xmlPathDoc = new  XPathDocument(xmlFileName);

// create a navigator for the xpath doc
XPathNavigator xNav = xmlPathDoc.CreateNavigator();

result = cxXPath.CreateXmlNodes(xNav, xmlDoc, 1, false, 1);

Returns a CxList of nodes in a given CxXmlDoc.
```

#### Version Information

Supported from 8.6.0

### 5.172.3 CxList FindXmlNodesByQualifiedName Method (string xmlFilterFiles, int language, string prefix, string nodeName, bool includeAttributes, string attributeName = "", string attributeValue = "", bool usesRegex = false, bool ignoreCase = false)

Return a CxList with CxXmlNode elements following the values defined by the parameters.

#### Syntax

```
CxQL
public CxList FindXmlNodesByQualifiedName(string xmlFilterFiles, int language,
string prefix, string nodeName, bool includeAttributes, string attributeName “”,
string attributeValue = “”, bool usesRegex = false, bool ignoreCase = false)
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The name of the prefix.

string

The name of the node.

bool

Specifies if the search should include attributes. This field is not required and its default value is false.

string

The name of the attribute. This field is not required.

string

The value of the attribute. This field is not required.

bool

Specifies if the search should use regex. This field is not required and its default value is false.

bool

Specifies if the search should be case sensitive. This field is not required and its default value is false.

**Return Value**

Return a CxList with CxXmlNode elements.

#### Exceptions

| Exception type | Condition |
|---|---|
| | |

#### Example

```
CxQL

The input source code is:
 <aura:attribute name="href" type="String" default="null"/>

result = cxXPath.FindXmlNodesByQualifiedName("*Test.app",8, "aura", "type");
// Result now holds the entire tag block
```

### Version Information

Supported from 8.6.0

## 5.172.4 CxList FindXmlNodesByQualifiedNameAndValue Method (string xmlFilterFiles, int language, string prefix, string nodeName, string nodeValue, bool usesRegexForNodeValue = false, bool ignoreCase = false)

Returns a CxList with CxXmlNode elements that contain the same name and value defined in the parameters.

### Syntax

```
CxQL
public CxList FindXmlNodesByQualifiedNameAndValue(string xmlFilterFiles, int
language, string prefix, string nodeName, string nodeValue, bool
usesRegexFornodeValue = false, bool ignoreCase = false)
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The name of the prefix.

string

The name of the node.

string

The value of the node.

bool

Specifies if the search should use regex for the node value. This field is not required and its default value is false.

bool

Specifies if the search should be case sensitive. This field is not required and its default value is false.

**Return Value**

Return a CxList with CxXmlNode elements.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

The input source code is:
 <aura:attribute name="href" type="String" default="null">
       {!userManager.isAuthorized}
 <aura:attribute/>


result = cxXPath. FindXmlNodesByQualifiedNameAndValue("*Test.app", 8, "aura",
"attribute", “userManager.isAuthorized”);
```

```
// Result now holds the entire tag block
```

### Version Information

Supported from 8.6.0

## 5.172.5 CxList FindXmlNodesByLocalName Method (string xmlFilterFiles, int language, string nodeName, bool includeAttributes = false, string attributeName = "", string attributeValue = "", bool usesRegex = false, bool ignoreCase = false)

Returns a CxList with CxXmlNode elements that contain the same local name defined in the parameters.

### Syntax

```
CxQL
public CxList FindXmlNodesByLocalName(string xmlFilterFiles, int language, string
nodeName, bool includeAttributes = false, string attributeName = ””, string
attributeValue = ””, bool usesRegex = false, bool ignoreCase = false)
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The name of the node.

bool

Specifies whether the search includes the attributes. This field is not required and its default value is false.

string

The name of the attribute. This field is not required.

string

The value of the attribute. This field is not required.

bool

Specifies if the search should use regex. This field is not required and its default value is false.

bool

Specifies if the search should be case sensitive. This field is not required and its default value is false.

**Return Value**

Return a CxList with CxXmlNode elements.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL


The input source code is:
 <a href="{!obj.href}">click me</a>

result = cxXPath. FindXmlNodesByLocalName("*.cmp", 8, "a", true, "href", 
"[{][!][^}]+[}]", true, true);

// Result now holds the entire tag block
```

### Version Information

Supported from 8.6.0

## 5.172.6 CxList FindXmlNodesByLocalNameAndValue Method (string xmlFilterFiles, int language, string nodeName, string nodeValue, bool usesRegexForNodeValue = false, bool ignoreCase = false)

Returns a CxList with CxXmlNode elements that contain the same local name and value defined in the parameters.

### Syntax

```
CxQL
public CxList FindXmlNodesByLocalNameAndValue(string xmlFilterFiles, int 
language, string nodeName, string nodeValue, bool usesRegexForNodeValue = false, 
bool ignoreCase = false)
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The name of the node.

string

The value of the node.

bool

Specifies if the search should use regex for the node value. This field is not required and its default value is false.

bool

Specifies if the search should be case sensitive. This field is not required and its default value is false.

**Return Value**

Returns a CxList with CxXmlNode elements.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

The input source code is:
 <a href="{!obj.href}">click me</a>
```

```
result = cxXPath. FindXmlNodesByLocalNameAndValue("*.cmp", 8, "a", “click me”,
true, "href", "[{][!][^}]+[}]", true, true);

// Result now holds the entire tag block
```

### Version Information

Supported from 8.6.0

## 5.172.7 CxList FindXmlAttributesByName Method (string xmlFilterFiles, int language, string attributeName, bool ignoreCase = false)

Returns a CxList with CxXmlNode elements that contain attributes with the same name defined in the parameters.

### Syntax

```
CxQL
public CxList FindXmlAttributesByName(string xmlFilterFiles, int language, string
attributeName, bool ignoreCase = false )
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The name of the attribute.

bool

Specifies if the search should be case sensitive. This field is not required and its default value is false.

**Return Value**

Return a CxList with CxXmlNode elements.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

The input source code is:
 <div id="error-section"></div>

result = cxXPath. FindXmlAttributesByName("*.app", 8, "id", true);

// Result now holds the entire tag block
```

### Version Information

Supported from 8.6.0

### 5.172.8 CxList FindXmlAttributesByValue Method (string xmlFilterFiles, int language, string attributeValue, bool usesRegex = false)

Returns a CxList with CxXmlNode elements that contain attributes with the same value defined in the parameters.

#### Syntax

```
CxQL
public CxList FindXmlAttributesByValue(string xmlFilterFiles, int language,
string attributeValue, bool usesRegex = false)
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The value of the attribute.

bool

Specifies if the search should use regex.

**Return Value**

Return a CxList with CxXmlNode elements.

#### Exceptions

| Exception type | Condition |
|---|---|
| | |

#### Example

```
CxQL

The input source code is:
 <div id="error-section"></div>

result = cxXPath.FindXmlAttributesByValue("*.app",8,"error-section", false);

// Result now holds the entire tag block
```

#### Version Information

Supported from 8.6.0

### 5.172.9 CxList FindXmlAttributesByNameAndValue Method (string xmlFilterFiles, int language, string attributeName, string attributeValue, bool usesRegex = false, bool ignoreCase = false)

Returns a CxList with CxXmlNode elements that contain attributes with the same name and value defined in the parameters.

#### Syntax

```
CxQL
```

```
public CxList FindXmlAttributesByNameAndValue(string xmlFilterFiles, int
language, string attributeName, string attributeValue, bool usesRegex = false,
bool ignoreCase = false)
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The name of the attribute.

string

The value of the attribute.

bool

Specifies if the search should use regex.

bool

Specifies if the search should be case sensitive. This field is not required and its default value is false.

**Return Value**

Return a CxList with CxXmlNode elements.

## Exceptions

| Exception type | Condition |
|---|---|
| | |

## Example

```
CxQL

The input source code is:
 <div id="error-section"></div>

result = cxXPath. FindXmlAttributesByNameAndValue("*.app", 8, “id”, "error-
section", false, true);

// Result now holds the entire tag block
```

## Version Information

Supported from 8.6.0

### 5.172.10 Void AddSupportForExpressionLanguageForFramework (string framework)

Sets the support for expression language for the framework to ture. Must be done at least once in a scan before running cxxPAth queries for expression language.

## Syntax

```
CxQL
public void AddSupportForExpressionLanguageForFramework (string framework)
```

**Parameters**

string

The name of the framework

## Exceptions

| Exception type | Condition |
|---|---|
| | |

## Remarks

2. This function is deprecated since 9.4.0.

## Example

```
CxQL

The input source code is:
<aura:component >
    <aura:attribute name="src" type="String"
default="javascript:alert(document.cookie)"/>
                                                        Attribute test in CSP
mode (LEX)
    <iframe src="{!v.src}"></iframe>
</aura:component>

       cxXPath.AddSupportForExpressionLanguageForFramework("Lightning");
// will set the support of the lightning expression language to true
```

## Version Information

Supported from 8.9.0

### 5.172.11 public CxList FindAllAttributesThatHoldExpressions(string xmlFilterFiles, int language, string framework)

Returns a CxList with CxXmlNode elements that contain nodes that have hold expressions as an attribute value.

## Syntax

```
public CxList FindAllAttributesThatHoldExpressions(string xmlFilterFiles, int
language, string framework)
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The name of the framework

**Return Value**

Return a CxList with CxXmlNode elements.

## Exceptions

| Exception type | Condition |
|---|---|
| NullReferenceException | **parameter is a null reference** |

### Remarks

3. This function is deprecated since 9.4.0.

### Example

```
CxQL

<ui:button label="Get custom objects" press="{!c.get_objs}"/>
result=cxXPath.FindAllAttributesThatHoldExpressions("*.cmp", 8, "Lightning");
// Result now holds the attribute which value is an expression (“press” CxXml
node)
```

### Version Information

Supported from 8.9.0

## 5.172.12 public CxList GetTextNodesExpressions(string xmlFilterFiles, string framework, int language)

Returns a CxList with all the expressions that are located in XML text for a given framework

### Syntax

```
public CxList GetTextNodesExpressions(string xmlFilterFiles, string
framework, int language)
```

**Parameters**

string

File extension pattern to be used as filter.

string

The name of the framework

int

Id of the language.

**Return Value**

A CxList which contains expressions that are located inside the XML text

### Exceptions

| Exception type | Condition |
|---|---|
| NullReferenceException | **parameter is a null reference** |

### Remarks

1. This function is deprecated since 9.4.0.

### Example

```
CxQL

<p><a href="{!obj.Phone}">Phone of {!obj.Name}</a></p>

result  =  cxXPath.GetTextNodesExpressions("*.cmp", "Lightning", 8);

// Result now holds the Member Access of “Name”
```

### Version Information

Supported from 8.9.0

### 5.172.13 public CxList createAttributesDefinition(string xmlFilterFiles, int language, string framework)

Returns a CxList with elements that will represent a Declarators of a framework specific variable definitions.

#### Syntax

```
public CxList createAttributesDefinition(string xmlFilterFiles, int language,
string framework)
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The name of the framework

**Return Value**

A CxList which contains a set of Declarators that are specific for a given framework

#### Exceptions

| Exception type | Condition |
|---|---|
| NullReferenceException | **parameter is a null reference** |

#### Remarks

1. This function is deprecated since 9.4.0.

#### Example

```
CxQL

    <aura:attribute name="accounts" type="Account[]"/>

result=cxXPath.createAttributesDefinition("*.cmp", 8, "Lightning");

// Result now holds Declarator of a variable of name “accounts” of type
“Account[]” and of private accessibility
```

#### Version Information

Supported from 8.9.0

### 5.172.14 public CxList GetXMLNodeDescendents(CxList originNodes, CxList descendentExpressionGroup)

Returns a CxList which is a subset of descendentExpressionGroup and is a XML DOM descendent of CxXML originNodes

#### Syntax

```
public CxList GetXMLNodeDescendents(CxList originNodes, CxList
descendentExpressionGroup)
```

**Parameters**

CxList

A list of CxXML elements which would serve as the scope to look under

CxList

A list of Expressions from which the Descendants will be extracted

**Return Value**

A CxList which is a subset of descendentExpressionGroup and is a XML DOM descendent of CxXML originNodes

## Exceptions

| Exception type | Condition |
|---|---|
| NullReferenceException | **parameter is a null reference** |

## Remarks

2. This function is deprecated since 9.4.0.

## Example

```
CxQL

    <aura:iteration var="obj" items="{!v.saccounts}">
    <p>
<a href="{!obj.Name}">{!'Name of ' + obj.Name}</a></p>
        <p><a href="{!obj.Phone}">Phone of {!obj.Name}</a></p>
    </aura:iteration>
CxList iteration = cxXPath.CreateIterationVarDefinition("*.cmp", 8,
"Lightning");
       CxList elementOfDeclarator =
cxXPath.GetElementOfCreatedDeclaration(iteration);
       CxList desc = cxXPath.GetXMLNodeDescendents(elementOfDeclarator,
allExpressionsInProject);

// Assumming that allExpressionsInProject will hold all expressions in the
framework, result will contain the expressions for obj.Name MemberAccesses ,
Obj.Phone MemberAccesses and a binaryExpression (in total 4 CxList results)
```

## Version Information

Supported from 8.9.0

### 5.172.15 public CxList GetAllExpressionDescendents(CxList descendentExpressionGroup, int language)

Returns a CxList which is all descendants of the descendentExpressionGroup

## Syntax

```
public CxList GetAllExpressionDescendents(CxList descendentExpressionGroup, int
language)
```

**Parameters**

CxList

The Ancestors whose descendants are to be returned int

Id of the language.

**Return Value**

Returns all elements that descends any of the elements in the descendentExpressionGroup parameter

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

<iframe src="{!'https:' + v.frameSrc}">iframe</iframe>

CxList attr=cxXPath.FindAllAttributesThatHoldExpressions("*.cmp", 8,
"Lightning");
CxList expr=cxXPath.GetExpressionsByAttributes(attr);
result=cxXPath.GetAllExpressionDescendents(expr,8);
// will return 4 results (binaryExpr, StringLiteral, MemberAccess,
UnknownReference).
```

### Version Information

Supported from 8.9.0

## 5.172.16 public CxList GetElementOfCreatedDeclaration(CxList declarations)

Returns a CxList with CxXmlNode elements that are DOM related to framework created Declarators

### Syntax

```
public CxList GetElementOfCreatedDeclaration(CxList declarations)
```

**Parameters**

CxList

A list of XML framework related Declarators

**Return Value**

A CxList with CxXmlNode elements that are DOM related to framework created Declarators

### Exceptions

| Exception type | Condition |
|---|---|
| NullReferenceException | **parameter is a null reference** |

### Remarks

1. This function is deprecated since 9.4.0.

### Example

```
CxQL

    <aura:attribute name="Id" type="String" access="Private" default="nil"/>

CxList declarators=cxXPath.createAttributesDefinition("*.cmp", 8, "Lightning");
result=cxXPath.GetElementOfCreatedDeclaration(declarators);

// result will contain “attribute” CxXmlNode element
```

### Version Information

Supported from 8.9.0

## 5.172.17 public CxList GetAttributeByExpression(CxList expression)

Returns a CxList with CxXmlNode attributes that are DOM related (key of expression) to the expressions given as a parameter

### Syntax

```
public CxList GetAttributeByExpression(CxList expression)
```

**Parameters**

CxList

A list of expressions that appear as a value to an XML attribute

**Return Value**

A CxList with CxXmlNode attributes that are DOM related (key of expression) to the expressions given as a parameter

### Exceptions

| Exception type | Condition |
|---|---|
| NullReferenceException | **parameter is a null reference** |

### Remarks

2. This function is deprecated since 9.4.0.

### Example

```
CxQL

    <iframe src="{!v.src}"></iframe>

CxList AllAttributesThatHoldExpressions =
cxXPath.FindAllAttributesThatHoldExpressions("*.cmp", 8, "Lightning");
CxList expression =
cxXPath.GetExpressionsByAttributes(AllAttributesThatHoldExpressions);
result = cxXPath.GetAttributeByExpression(expression);

// result will contain one CxXML node for “src” attribute
```

### Version Information

Supported from 8.9.0

### 5.172.18 public CxList GetExpressionsByAttributes(CxList attributes)

Returns a CxList of expressions that are DOM related to the attributes parsed as param (given a key of attributes will return a set of expressions that are their values).

#### Syntax

```
public CxList GetExpressionsByAttributes(CxList attributes)
```

**Parameters**

CxList

A list of attributes that have an expression as a vlue

**Return Value**

A CxList of expressions that are DOM related to the attributes parsed as param

#### Remarks

1. This function is deprecated since 9.4.0.

#### Exceptions

| Exception type | Condition |
|---|---|
| NullReferenceException | **parameter is a null reference** |

#### Example

```
CxQL

    <iframe src="{!v.src}"></iframe>

CxList AllAttributesThatHoldExpressions =
cxXPath.FindAllAttributesThatHoldExpressions("*.cmp", 8, "Lightning");
Result= cxXPath.GetExpressionsByAttributes(AllAttributesThatHoldExpressions);

// result will return a MemberAccess “src”
```

#### Version Information

Supported from 8.9.0

### 5.172.19 public CxList GetElementByExpression(CxList expressions)

Returns a CxList of CxXmlNode that hold the element that is DOM related to a given expression.

#### Syntax

```
public CxList GetElementByExpression(CxList expressions)
```

**Parameters**

CxList

A list of expressions

**Return Value**

A CxList of CxXmlNode that hold the element that is DOM related to a given expression

### Exceptions

| Exception type | Condition |
|---|---|
| NullReferenceException | **parameter is a null reference** |

### Remarks

1. This function is deprecated since 9.4.0.

### Example

```
CxQL

    <iframe src="{!v.src}"></iframe>

CxList AllAttributesThatHoldExpressions =
cxXPath.FindAllAttributesThatHoldExpressions("*.cmp", 8, "Lightning");
CxList exp=
cxXPath.GetExpressionsByAttributes(AllAttributesThatHoldExpressions);
result= cxXPath.GetElementByExpression(exp);

// result will contain the CxXMlNode for iframe element.
```

### Version Information

Supported from 8.9.0

## 5.172.20 public CxList CreateIterationVarDefinition(string xmlFilterFiles, int language, string framework)

Returns a CxList with elements that will represent a Declarators of a framework specific iteration statement.

### Syntax

```
public CxList CreateIterationVarDefinition(string xmlFilterFiles, int language,
string framework)
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The name of the framework

**Return Value**

A CxList which contains a set of Declarators that are specific for a given framework

### Exceptions

| Exception type | Condition |
|---|---|
| NullReferenceException | **parameter is a null reference** |

### Remarks

1. This function is deprecated since 9.4.0.

### Example

```
CxQL

    <aura:iteration var="obj" items="{!v.customobjects}">

result=cxXPath.CreateIterationVarDefinition ("*.cmp", 8, "Lightning");

// Result now holds Declarator of a variable of name “obj” of type relevant to
customobjects and of private accessibility
```

### Version Information

Supported from 8.9.0

## 5.173 CxList CxJson Methods

### 5.173.1 CxList FindJsonPropertyByName Method (string filesFilter, int language, string name, bool usesRegex = false, bool ignoreCase = false)

Returns a CxList with CxJsonProperty elements following the values defined by the parameters.

### Syntax

```
CxQL
public CxList FindJsonPropertyByName (string filesFilter, int language, string
name, bool usesRegex  = false, bool ignoreCase = false)
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The name of the property.

bool

Specifies whether the search should use regex. This field is not required and its default value is false.

bool

Specifies whether the search should be case sensitive. This field is not required and its default value is false.

**Return Value**

Returns a CxList with CxJsonProperty elements

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

{
  "proxies": {
    "default": {
      "httpProxy": "http://10.0.50.1:3128",
      "httpsProxy":  "http://10.0.50.1:3128"
    }
  }
}

result=cxJson.FindJsonPropertyByName("config.json", 8, "proxy$", true, true);

// Result now holds two CxJsonProperty
```

### Version Information

Supported from 9.2.0

## 5.173.2 CxList FindJsonPropertyByValue Method (string filesFilter, int language, string value, bool usesRegex = false, bool ignoreCase = false)

Returns a CxList with CxJsonProperty elements following the values defined by the parameters.

### Syntax

```
CxQL
public CxList FindJsonPropertyByValue (string filesFilter, int language, string
value, bool usesRegex  = false, bool ignoreCase = false)
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The value of the property.

bool

Specifies whether the search should use regex. This field is not required and its default value is false.

bool

Specifies whether the search should be case sensitive. This field is not required and its default value is false.

**Return Value**

Returns a CxList with CxJsonProperty elements

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

{
  "proxies": {
    "default": {
      "httpProxy": "http://10.0.50.1:3128",
      "httpsProxy":  "http://10.0.50.1:3128"
    }
  }
}

result=cxJson.FindJsonPropertyByValue("*.json", 8, "^http", true, false);

// Result now holds two CxJsonProperty
```

### Version Information

Supported from 9.2.0

## 5.173.3 CxList FindJsonPropertyByNameAndValue Method (string filesFilter, int language, string name, string value, bool usesRegex = false, bool ignoreCase = false)

Returns a CxList with CxJsonProperty elements following the values defined by the parameters.

### Syntax

```
CxQL
public CxList FindJsonPropertyByNameAndValue (string filesFilter, int language,
string name, string value, bool usesRegex  = false, bool ignoreCase = false)
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The name of the property.

string

The value of the property.

bool

Specifies whether the search should use regex. This field is not required and its default value is false.

bool

Specifies whether the search should be case sensitive. This field is not required and its default value is false.

**Return Value**

Returns a CxList with CxJsonProperty elements

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

{
  "proxies": {
    "default": {
      "httpProxy": "http://10.0.50.1:3128",
      "httpsProxy":  "http://10.0.50.1:3128"
    }
  }
}

result=cxJson.FindJsonPropertyByNameAndValue("*.json", 8, "^https", “^http:”,
true, false);

// Result now holds one CxJsonProperty
```

### Version Information

Supported from 9.2.0

## 5.173.4 CxList FindChildPropertiesByJsonPath Method (string filesFilter, int language, string path)

Returns a CxList with CxJsonProperty elements following the values defined by the parameters.

### Syntax

```
CxQL
public CxList FindChildPropertiesByJsonPath(string filesFilter, int language,
string path)
```

**Parameters**

string

File extension pattern to be used as filter.

int

Id of the language.

string

The path in the Json file based on the JsonPath expression.

**Return Value**

Returns a CxList with CxJsonProperty elements.

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

{
  "proxies": {
    "rootpath": {
      "httpsProxy":  "http://10.0.50.1:3128"
    },
    "mypath": "http://10.0.50.1:8080"
  }
}

CxList res = cxJson.FindChildPropertiesByJsonPath ("*.json", 2, "$.proxies");
// Result now holds the rootpath and the mypath value (http://10.0.50.1:8080)
```

### Version Information

Supported from 9.2.0

# 6 Method Documentation (for internal use only)

## 6.1 CxList.SetQueryProperty Method (QueryProperties.propertyEnum, QueryProperties.flowDirectionEnum)

Adds/modifies a query property of the current query

### Syntax

```
CxQL
public static void SetQueryProperty (QueryProperties.propertyEnum queryProperty,
QueryProperties.flowDirectionEnum directionValue)
```

**Parameters**

**queryProperty**

enum of query properties: `FLOW_DIRECTION`

**directionValue**

enum of flow direction: `From_Small_To_Large = 1, From_Large_To_Small, From_Start, From_End`

### Exceptions

| Exception type | Condition |
|---|---|
| | |

### Example

```
CxQL

SetQueryProperty(QueryProperties.propertyEnum.FLOW_DIRECTION,
QueryProperties.flowDirectionEnum.From_Start); //makes the calculation of flows
be from start node to end node.
```

### Version Information

Supported from 8.0.0

## 6.2 CxList.GetSanitizerByMethodInCondition Method (CxList)

For each input method, finds all the calls inside a "if" condition and returns all the references, of the methods parameters, that are inside the "if" statement.

### Syntax

```
CxQL
public CxList GetSanitizerByMethodInCondition(CxList MethodCallsInCondition)
```

**Parameters**

**MethodCallsInCondition**

method call list inside "if" condition (must be of type MethodInvoke)

**Return Value**

all references of a method call parameter in the scope of the if statement

### Example

```
This example demonstrates the CxList.GetSanitizerByMethodInCondition(CxList
MethodCallsInCondition) method.
 The input source code is:
       String a = getInput();
       if(Valid(a)){
              Print(a);
       }

CxList valid = All.FindByShortName("Valid");
result = All.GetSanitizerByMethodInCondition(valid);

the result would consist of 1 item:
            a (in Print(a))
The purpose of the query is to mark 'a' as a sanitizer, because the flow doesn't
pass through the condition.
```

#### Remarks

Calls GetSanitizerByMethodInCondition(MethodCallsInCondition, IfBlock.Both).

### Version Information

Supported from 7.1.2

## 6.3 CxList.GetSanitizerByMethodInCondition Method (CxList, IfBlock)

For each method, finds all the calls inside a "if" condition and returns all the references, of the methods parameters, that are inside the "if" block (the IfBlock input parameter)  statement.

### Syntax

CxQL
```
public CxList GetSanitizerByMethodInCondition(CxList MethodCallsInCondition,
IfBlock block)
```

**Parameters**

**MethodCallsInCondition**

method call list inside "if" condition (must be of type MethodInvoke)

**block**

select only "true", only "false" or both scopes.

**Return Value**

all references of a method call parameter in the scope of the if statement

### Example

```
This example demonstrates the CxList.GetSanitizerByMethodInCondition(CxList
MethodCallsInCondition, IfBlock block) method.
 The input source code is:
       String a = getInput();
       if(Valid(a)){
              Print(a);
       }

CxList valid = All.FindByShortName("Valid");
```

```
result = All.GetSanitizerByMethodInCondition(valid,CxList.IfBlock.True);

the result would consist of 1 item:
            a (in Print(a))
The purpose of the query is to mark 'a' as a sanitizer, because the flow doesn't
pass through the condition.
```

## Version Information

Supported from 7.1.2

# 7 CxList Operators

The operators of the **CxList** class are listed here.
Public Operators

| | |
|---|---|
| == Operator | Determines whether two specified CxList objects have the same values. |
| != Operator | Determines whether two specified CxList objects do not have the same values (they differ by a least one value). |
| + Operator | Merges two specified CxList objects (same as \| operator) |
| - Operator | Removes the values of second specified CxList object from the first one. |
| & Operators | Intersects the values of the two specified CxList objects (same as * operator) |
| \| Operator | Merges two specified CxList object (same as + operator) |
| * Operator | Intersects the values of the two specified CxList objects (same as & operator) |
| < Operator | Return true if the first specified CxList is completely contained within the second one. |
| > Operator | Return true if the second specified CxList is completely contained within the first one. |
| <= Operator | Return true if the first specified CxList is completely contained within the second one or they are equal. |
| => Operator | Return true if the second specified CxList is completely contained within the first one or they are equal. |

# 8 CxQuery Miscellaneous Methods

## 8.1 cxLog.WriteDebugMessage Method (string)

Display string to DebugMessages tab in CxAudit program

### Syntax

```
CxQL
public void cxLog.WriteDebugMessage(string str)
```

**Parameters**

String to be displayed.

**Return Value**

none.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | **parameter is a null reference** |

### Remarks

All calling to cxLog.WriteDebugMessage should be removed from production version!!!

It also prints the debug message into the CxSAST log file (even outside CxAudit).

### Example

The following code example shows how you can use the cxLog.WriteDebugMessage method.

```
CxQL

This example demonstrates the cxLog.WriteDebugMessage method.
The input source code is:
 class cl1
 {
      void foo()
      {
             int a = 3;
             int b = 5;
      }
 }
result = All.FindByShortName("foo");
if (result.Count > 0)
       cxLog.WriteDebugMessage(result.GetFirstGraph().ShortName);
       cxLog.WriteDebugMessage (“number of DOM elements =” + All.Count);

the result would be – on DebugMessage tab in CxAudit program
       foo
       number of DOM elements = 14
```

### Version Information

Supported from v1.8.1

# 9 Appendix A. CxDOM Types

The built-in types in CxDOM are listed here:

| Types | Types (cont.) | Types(Cont.) |
|---|---|---|
| AccessorDecl | DelegateInvokeExpr | OperatorDecl |
| AccessorModifiers | DestructorDecl | OverloadableOperator |
| ArgumentRef | DictionaryCreateExpr | Param |
| ArrayCreateExpr | DictionaryInitializer | ParamDecl |
| ArrayElementRef | EnumDecl | ParamDirection |
| ArrayInitializer | EnumMemberDecl | PointerTypeRef |
| AssemblyReference | EventDecl | PostfixExpr |
| AssignExpr | EventRef | PostfixOperator |
| AssignOperator | ExprStmt | PropertyDecl |
| AssociativeArrayExpr | FieldDecl | PropertyRef |
| AssociativeArrayPropertyEntry | FieldRef | PropertySetValueRef |
| AssociativeArrayRegularEntry | ForEachStmt | RankSpecifier |
| AttachDelegateStmt | FunctionTypeRef | RealLiteral |
| AttributeTarget | GenericsConstraints | RemoveDelegateStmt |
| BaseRef | GenericTypeRef | ReturnStmt |
| BinaryExpr | GotoStmt | SliceExpression |
| BinaryOperator | GraphTypes | StringLiteral |
| BooleanLiteral | IfStmt | StructDecl |
| BreakStmt | Import | SubExpr |
| BuiltInType | IndexerDecl | SwitchStmt |
| Case | IndexerRef | TernaryExpr |
| CastExpr | IntegerLiteral | ThisRef |
| Catch | InterfaceDecl | ThrowStmt |
| CharLiteral | IterationStmt | TryCatchFinallyStmt |
| CheckedStmt | IterationType | TupleCreateExpr |
| ClassDecl | LabeledStmt | TupleInitializer |
| CodeCollectionAttribute | LambdaExpr | TypeKind |
| CodeElementAttribute | LinePragma | TypeModifiers |
| Comment | LocalRef | TypeOfExpr |
| CommentStmt | LockStmt | TypeRef |
| ConstantDecl | MemberAccess | UnaryExpr |
| ConstantDeclStmt | MemberKind | UnaryOperator |
| ConstructorDecl | MethodDecl | UncheckedStmt |
| ContinueStmt | MethodInvokeExpr | UnknownReference |
| CreateDelegateExpr | MethodRef | UsingStmt |
| CustomAttribute | Modifiers | VariableDecl |
| CxXmlNode | NamespaceDecl | VariableDeclStmt |
| Declarator | NullLiteral | |

| Types | Types (cont.) | Types(Cont.) |
|---|---|---|
| DelegateDecl | ObjectCreateExpr | |

## Example

In order to better understand each of these types, try the following query:

```
CxQL

result = All.FindByType(typeof(IfStmt));
"IfStmt" is one of the above types.
```