Syntax for the Work Item Query Language (WIQL)

07/04/2019 • 12 minutos para ler • Colaboradores

Neste artigo

Example queries
Fields
Operators
Logical grouping
Macros
Historical queries (ASOF)
Sorting results (ORDER BY)
Related articles

Azure Boards | Azure DevOps Server 2019 | TFS 2018 | TFS 2017 | TFS 2015 | TFS 2013

You can use the WIQL syntax to define a query as a hyperlink or when using the Work Item Query Language (REST API).

A query defined using the Work Item Query Language (WIQL) consists of a `SELECT` statement that lists the fields to be returned as columns in the result set. You can further qualify the result set by using a logical expression. You can specify a sort order. Use an `ASOF` clause to state that a query is evaluated as of a previous time.

The WIQL syntax is not case sensitive.

Example queries

The following typical WIQL query example uses reference names for the fields. The query selects work items (no work item type specified) with a **Priority=1**. The query returns the **ID** and **Title** of the return set as columns. The results are sorted by **ID** in ascending order.

WIQL Copiar

```
SELECT System.ID, System.Title from workitems
where Priority=1 order by System.ID asc
```

The general format of a query consists of a `SELECT` statement and a qualifying `WHERE` clause. The following examples illustrate the basic syntax.

WIQL Copiar

```
SELECT Select_List
   FROM workitems
   [WHERE Conditions]
   [ORDER BY Column_List]
   [ASOF DateTimeConditions]
```

Date-time pattern

You specify the date-time pattern according to one of two patterns:

- The Date Pattern and Time Pattern you set under your personal profile settings (Set personal preferences).
- The pattern specified by UTC which follows this pattern (with Z appended to the date-time):
  `AND System.ChangedDate >= '1/1/2019 00:00:00Z'`

Example clauses

The following example statements show specific qualifying clauses.

Clause	Example

`AND`	Copiar
	`SELECT [System.Id], [System.Title]`
	`FROM WorkItems`
	`WHERE [System.TeamProject] = @project`
	`AND [System.AssignedTo] = 'Judy Lew'`

`OR`	Copiar
	`SELECT [System.Id], [System.Title]`
	`FROM WorkItems`
	`WHERE [System.TeamProject] = @project`
	`AND ( [System.AssignedTo] = 'Mark Steele'`
	`OR [System.AssignedTo] = 'Merav Sror' )`

`NOT`	Copiar
	`SELECT [System.Id], [System.Title]`
	`FROM WorkItems`
	`WHERE [System.TeamProject] = @project`
	`AND [System.AssignedTo] EVER 'Anne Wallace'`
	`AND [System.AssignedTo] NOT CONTAINS 'Danny Levin'`

`EVER`	Copiar
	`SELECT [System.Id], [System.Title]`
	`FROM WorkItems`
	`WHERE [System.TeamProject] = @project`
	`AND [System.AssignedTo] EVER 'Anne Wallace'`

`UNDER`	Copiar

```
SELECT [System.Id], [System.Title]
   FROM WorkItems
   WHERE [System.TeamProject] = @project
   AND [System.AssignedTo] EVER 'David Galvin'
   AND [System.AreaPath] UNDER 'Agile1\Area 0'
```

Sort (`ORDER`)

Copiar

```
SELECT [System.Id], [System.Title]
   FROM WorkItems
   WHERE [System.TeamProject] = @project
   AND [System.AssignedTo] = 'Jon Ganio'
   ORDER BY [System.Id] [asc | desc]
```

Time filter (`ASOF`)

Copiar

```
SELECT [System.Title]
   FROM workitems
   WHERE [System.IterationPath] = 'MyProject\Beta'
   AND [System.AssignedTo] = 'Jim Daly'
   ASOF '3/16/06 12:30'
```

Fields

When specifying fields, you can use the reference name or friendly name. The following examples are valid WIQL syntax.

Supported field references	Example
Using reference name with spaces	Copiar `SELECT [System.AssignedTo] FROM ...`
Using a friendly name with spaces	Copiar `SELECT [Assigned To] FROM ...`
Using names without spaces, no square brackets required	Copiar `SELECT ID, Title FROM ...`

Operators

Queries use logical expressions to qualify result sets. These logical expressions are formed by one or more conjoined operations.

Some simple query operations are listed below.

WIQL Copiar

```
WHERE [System.AssignedTo] = 'joselugo'
WHERE [Adatum.CustomMethodology.Severity] >= 2
```

The table below summarizes all the supported operators for different field types. For additional information on each field type, see Work item fields and attributes.

The `=, <>, >, <, >=, and <=` operators work as expected. For instance, `System.ID > 100` queries for all work items with an **ID** greater than 100. `System.ChangedDate > '1/1/16 12:00:00'` queries for all work items changed after noon of January 1, 2016.

Beyond these basic operators, there are some behaviors and operators specific to certain field types.

> **Observação**
>
> The operators available to you depend on your platform and version. For more information, see Query quick reference.

Field type	Supported operators
Boolean	`= , <> , =[Field] , <>[Field]`
Double, GUID, Integer	`= , <> , > , < , >= , <= , =[Field], <>[Field], >[Field], <[Field], >=[Field], <=[Field], In, Not In, Was Ever`
Boolean	`= , <> , =[Field] , <>[Field]`
DateTime	`= , <> , > , < , >= , <= , =[Field], <>[Field], >[Field], <[Field], >=[Field], <=[Field], In, Not In, Was Ever`
Identity	`= , <> , > , < , >= , <= , =[Field], <>[Field], >[Field], <[Field], >=[Field], <=[Field], Contains, Does Not Contain, In, Not In, In Group, Not In Group, Was Ever`
PlainText	`Contains Words, Does Not Contain Words, Is Empty, Is Not Empty`
String	`= , <> , > , < , >= , <= , =[Field], <>[Field], >[Field], <[Field], >=[Field], <=[Field], Contains, Does Not Contain, In, Not In, In Group, Not In Group, Was Ever`
TreePath	`=, <>, In, Not In, Under, Not Under`

DateTime

You must quote (single or double quotes are supported) DateTime literals used in comparisons. They must be in the .NET DateTime format of the local client computer running the query. Unless a time zone is specified, DateTime literals are in the time zone of the local computer.

WIQL Copiar

```
WHERE [Adatum.Lite.ResolvedDate] >= '1/8/06 GMT' and [Resolved Date/Time] < '1/9/06 GMT'
WHERE [Resolved Date] >= '1/8/06 14:30:01'
```

When the time is omitted in a DateTime literal and the dayPrecision parameter equals false, the time is assumed to be zero (midnight). The default setting for the dayPrecision parameter is false.

String and PlainText

You must quote string literals (single or double quotes are supported) in a comparison with a string or plain text field. String literals support all Unicode characters.

WIQL Copiar

```
WHERE [Adatum.Lite.Blocking] = 'Not Blocking'
WHERE [Adatum.Lite.Blocking] <> 'Blocked'
```

You can use the contains operator to search for a substring anywhere in the field value.

WIQL Copiar

```
WHERE [System.Description] contains 'WIQL'
```

Area and Iteration (TreePath)

You can use the under operator for the Area and Iteration Path fields. under evaluates whether a value is within the sub-tree of a specific classification node. For instance, the expression below would evaluate to true if the Area Path were 'MyProject\Server\Administration', 'MyProject\Server\Administration\Feature 1', 'MyProject\Server\Administration\Feature 2\SubFeature 5', or any other node within the sub-tree.

WIQL Copiar

```
WHERE [System.AreaPath] under 'MyProject\Server\Administration'
```

Modifiers and special operators

You can use some modifiers and special operators in a query expression.

Use the `in` operator to evaluate whether a field value is equal to any of a set of values. This operator is supported for the String, Integer, Double, and DateTime field types. See the following example along with its semantic equivalent.

WIQL Copiar

```
WHERE [System.CreatedBy] in ('joselugo', 'jeffhay', 'linaabola')
WHERE [System.CreatedBy] = 'joselugo' OR [System.CreatedBy] = 'jeffhay' OR [System.CreatedBy] =
'linaabola'
```

The ever operator is used to evaluate whether a field value equals or has ever equaled a particular value throughout all past revisions of work items. The String, Integer, Double, and DateTime field types support this operator. There are alternate syntaxes for the ever operator. For example, the snippets below query whether all work items were ever assigned to 'joselugo'.

WIQL Copiar

```
WHERE ever ([Assigned To] =  'joselugo')
WHERE [Assigned To] ever  'joselugo'
```

Logical grouping

You can use the terms `and` and `or` in the typical Boolean sense to evaluate two clauses. You can group logical expressions and further conjoin them, as needed. Examples are shown below.

WIQL Copiar

```
WHERE [System.State] =  'Active' and [System.AssignedTo] = 'joselugo' and ([System.CreatedBy] =
'linaabola'
    OR [Adatum.CustomMethodology.ResolvedBy] = 'jeffhay')
    AND [System.State] = 'Closed'
    WHERE [System.State] = 'Active'
    AND [System.State] EVER 'Closed'
```

You can negate the `contains, under,` and `in` operators by using `not`. You can't negate the `ever` operator. The examples below query for all work items that are not classified within the sub-tree of 'MyProject\Feature1'.

WIQL Copiar

```
WHERE [System.AreaPath] not under 'MyProject\Feature1'
WHERE [System.AssignedTo] ever 'joselugo'
```

Macros

The following table lists the macros or variables you can use within a WIQL query.

Macro	Usage
@Me	Use this variable to automatically search for the current user's alias in a field that contains user aliases. For example, you can find work items that you opened if you set the **Field** column to **Activated By**, the **Operator** column to =, and the **Value** column to **@Me**.
@CurrentIteration	Use this variable to automatically filter for work items assigned to the current sprint for the selected team based on the selected team context.
@Project	Use this variable to search for work items in the current project. For example, you can find all the work items in the current project if you set the **Field** column to **Team Project**, the **Operator** column to =, and the **Value** column to **@Project**.
@StartOfDay **@StartOfWeek** **@StartOfMonth** **@StartOfYear**	Use these macros to filter DateTime fields based on the start of the current day, week, month, year or an offset to one of these. For example, you can find all items created in the last 3 months if you set the **Field** column to **Created Date**, the **Operator** column to >=, and the **Value** column to **@StartOfMonth - 3**.
@Today	Use this variable to search for work items that relate to the current date or to an earlier date. You can also modify the **@Today** variable by subtracting days. For example, you can find all items activated in the last week if you set the **Field** column to **Activated Date**, the **Operator** column to >=, and the **Value** column to **@Today - 7**.

Macro	Usage
[Any]	Use this variable to search for work items that relate to any value that is defined for a particular field.

> **Observação**
>
> Both the `@me` and `@today` macros have default values.

@me macro

The `@me` macro substitutes the Windows Integrated account name of the user who runs the query. The example below shows how to use the macro and the equivalent static statement. Although the macro is intended for fields such as `Assigned To`, you can use it for any String field, although the result may not be meaningful.

WIQL Copiar

```
[System.AssignedTo] = @Me
[System.AssignedTo] = 'joselugo'
```

@today macro

You can use the `@today` macro with any **DateTime** field. This macro substitutes midnight of the current date on the local computer that runs the query. You can also specify `@today+x` or `@today-y` using integer offsets for x days after `@today` and y days before `@today`, respectively. Note that a query that uses the `@today` macro can return different result sets depending on the time zone in which it is run.

The examples below assumes that today is 1/3/19.

WIQL Copiar

```
[System.CreatedDate] = @today
```

is the equivalent of:

WIQL Copiar

```
[System.CreatedDate] = '1/3/19'
```

and

WIQL Copiar

```
[System.CreatedDate] > @today-2
```

is the equivalent of:

WIQL Copiar

```
[System.CreatedDate] > '1/1/19'
```

@StartOfDay, @StartOfWeek, @StartOfMonth, @StartOfYear macros

You can use the `@StartOf...` macros with any **DateTime** field. This macro substitutes midnight of the current day, start of week, start of month, or start of year on the local computer that runs the query.

These macros accept a modifier string which has a format of `(+/-)nn(y|M|w|d|h|m)`. Similar to the @Today macro, you can specify plus or minus integer offsets. If the time unit qualifier is omitted, it defaults to the natural period of the function, e.g. `@StartOfWeek("+1")` is the same as `@StartOfWeek("+1w")`. If the plus/minus (+/-) sign is omitted, plus is assumed.

This syntax allows you to nest modifiers and offset your query twice. For example, the following clause filters work items that have been closed last year and three months into the start of the current year.

WIQL Copiar

```
[System.ClosedDate] >=@StartOfYear('+3M') - 1
```

The following examples assume that today is 4/5/19.

WIQL Copiar

```
[System.CreatedDate] >= @StartOfMonth-3
```

is the equivalent of:

WIQL Copiar

```
[System.CreatedDate] >= '1/1/19'
```

and

WIQL Copiar

```
[Microsoft.VSTS.Scheduling.TargetDate] > @StartOfYear
```

is the equivalent of:

WIQL Copiar

```
[Microsoft.VSTS.Scheduling.TargetDate]  > '1/1/19'
```

Custom macros

WIQL also supports arbitrary custom macros. Any string prefixed by an '@' is treated as a custom macro and will be substituted. The substitute value for the custom macro is retrieved from the context parameter of the query method in the object model. The following method is the API used for macros:

C# Copiar

```
public WorkItemCollection Query(string wiql, IDictionary context)
```

The context parameter contains key-value pairs for macros. For example, if the context contains a key-value pair of (project, MyProject), then '@project' will be replaced by 'MyProject' in the WIQL. This is how the work

item query builder handles the @project macro in Visual Studio.

Historical queries (ASOF)

You can use an `ASOF` clause in a query to filter for work items that satisfy the specified condition on a specific date at a specific time.

For example, suppose a work item was classified under an iteration path of MyProject\ProjArea and assigned to 'Mark Hanson' on 3/17/16. However, the work item was recently assigned to 'Roger Harui' and moved to a new iteration path of Release. The following example query will return this work item because the query is based on the state of work items as of a past date and time.

> **Observação**
>
> If no time is specified, WIQL uses midnight. If no time zone is specified, WIQL uses the time zone of the local client computer.

WIQL Copiar

```
SELECT [System.Title]
    FROM workitems
    WHERE ([System.IterationPath] = 'MyProject\ProjArea' and [System.AssignedTo] = 'Mark Han-
son')
    ASOF '3/16/16 12:30'
```

Sorting results (ORDER BY)

You can use the `ORDER BY` clause to sort the results of a query by one or more fields in ascending or descending order.

> **Observação**
>
> The sorting preferences of the SQL server on the data tier determine the default sort order. However, you can use the `asc` or `desc` parameters to choose an explicit sort order.

The following example sorts work items first by **Priority** in ascending order, and then by **Created Date** in descending order.

WIQL Copiar

```
SELECT [System.Title]
    FROM workitems
    WHERE [System.State] =  'Active' and [System.AssignedTo] =  'joselugo'
    ORDER BY [Microsoft.VSTS.Common.Priority] asc, [System.CreatedDate] desc
```

Related articles

- Query fields, operators, values, and variables
- Work item fields and attributes
- Wiql Editor, a Marketplace extension

Limits on WIQL length

For queries made against Azure Boards, the WIQL length must not exceed 32K characters. The system won't allow you to create or run queries that exceed that length.