ChilliCream / **graphql-platform** Public

Code | Issues 287 | Pull requests 89 | Discussions | Actions | Projects

Commit `4cbaf67`

michaelstaib authored last week · ✕ 96 / 98 · Verified

```
Add depth limit to GraphQL parser (#9528)
```

`main-version-15` (#9528) · 15.1.14

1 parent `7f8b04d` commit `4cbaf67`

13 files changed +364 -23 lines changed

Top

Filter files…

- src/HotChocolate
  - Core/src/Execution
    - DependencyInjection
      - RequestExecutorServiceCollectionExtensions.cs
    - Options
      - RequestParserOptions.cs
  - Language
    - src/Language.Utf8
      - ParserOptions.cs
      - Properties
        - LangUtf8Resources.Designer.cs
        - LangUtf8Resources.resx
      - Utf8GraphQLParser.Directives.cs
      - Utf8GraphQLParser.Fragments.cs
      - Utf8GraphQLParser.Operations.cs
      - Utf8GraphQLParser.Types.cs

Utf8GraphQLParser.Utilities.cs

Utf8GraphQLParser.Values.cs

Utf8GraphQLParser.cs

test/Language.Tests/Parser

QueryParserTests.cs

13 files changed +364 -23 lines changed

Search within code

...questExecutorServiceCollectionExtensions.cs

```
@@ -94,9 +94,12 @@ public static IServiceCollection AddGraphQLCore(this
IServiceCollection services
94  94
95  95                      return new ParserOptions(
96  96                          noLocations: !options.IncludeLocations,
    97  +                       allowFragmentVariables: false,
97  98                          maxAllowedNodes: options.MaxAllowedNodes,
98  99                          maxAllowedTokens: options.MaxAllowedTokens,
99      -                       maxAllowedFields: options.MaxAllowedFields);
    100 +                       maxAllowedFields: options.MaxAllowedFields,
    101 +                       maxAllowedDirectives: options.MaxAllowedDirectives,
    102 +                       maxAllowedRecursionDepth:
    options.MaxAllowedRecursionDepth);
100 103                 });
101 104
102 105             return services;
```

...c/Execution/Options/RequestParserOptions.cs

```
@@ -50,4 +50,17 @@ public sealed class RequestParserOptions
50  50          /// as fields is an easier way to estimate query size for GraphQL requests.
51  51          /// </summary>
52  52          public int MaxAllowedFields { get; set; } = 2048;
    53  +
    54  +       /// <summary>
    55  +       /// The maximum number of directives allowed per location (e.g. per field,
    56  +       /// per operation, per fragment definition). Repeatable directives can be
    used
```

```
57  +     /// to exhaust CPU and memory resources if not limited.
58  +     /// </summary>
59  +     public int MaxAllowedDirectives { get; set; } = 4;
60  +
61  +     /// <summary>
62  +     /// The maximum allowed recursion depth when parsing a document.
63  +     /// This prevents stack overflow from deeply nested queries.
64  +     /// </summary>
65  +     public int MaxAllowedRecursionDepth { get; set; } = 200;
53  66    }
```

...Language/src/Language.Utf8/ParserOptions.cs

```
@@ -45,6 +45,51 @@ public ParserOptions(
45  45            MaxAllowedTokens = maxAllowedTokens;
46  46            MaxAllowedNodes = maxAllowedNodes;
47  47            MaxAllowedFields = maxAllowedFields;
48  +         MaxAllowedDirectives = 4;
49  +         MaxAllowedRecursionDepth = 200;
50  +     }
51  +
52  +     /// <summary>
53  +     /// Initializes a new instance of <see cref="ParserOptions"/> with security limits.
54  +     /// </summary>
55  +     /// <param name="noLocations">
56  +     /// Defines that the parse shall not preserve syntax node locations.
57  +     /// </param>
58  +     /// <param name="allowFragmentVariables">
59  +     /// Defines that the parser shall parse fragment variables.
60  +     /// </param>
61  +     /// <param name="maxAllowedNodes">
62  +     /// The maximum number of nodes allowed within a document.
63  +     /// </param>
64  +     /// <param name="maxAllowedTokens">
65  +     /// The maximum number of tokens allowed within a document.
66  +     /// </param>
67  +     /// <param name="maxAllowedFields">
68  +     /// The maximum number of fields allowed within a query document.
69  +     /// </param>
70  +     /// <param name="maxAllowedDirectives">
```

```
                71  +     /// The maximum number of directives allowed per location (e.g. per field,
                72  +     /// per operation, per fragment definition).
                73  +     /// </param>
                74  +     /// <param name="maxAllowedRecursionDepth">
                75  +     /// The maximum allowed recursion depth of a parsed document.
                76  +     /// </param>
                77  +     public ParserOptions(
                78  +         bool noLocations,
                79  +         bool allowFragmentVariables,
                80  +         int maxAllowedNodes,
                81  +         int maxAllowedTokens,
                82  +         int maxAllowedFields,
                83  +         int maxAllowedDirectives,
                84  +         int maxAllowedRecursionDepth)
                85  +     {
                86  +         NoLocations = noLocations;
                87  +         Experimental = new(allowFragmentVariables);
                88  +         MaxAllowedTokens = maxAllowedTokens;
                89  +         MaxAllowedNodes = maxAllowedNodes;
                90  +         MaxAllowedFields = maxAllowedFields;
                91  +         MaxAllowedDirectives = maxAllowedDirectives;
                92  +         MaxAllowedRecursionDepth = maxAllowedRecursionDepth;
 48             93          }
 49             94
 50             95          /// <summary>
                    @@ -86,6 +131,18 @@ public ParserOptions(
 86            131          /// </summary>
 87            132          public int MaxAllowedFields { get; }
 88            133
                134  +     /// <summary>
                135  +     /// The maximum number of directives allowed per location (e.g. per field,
                136  +     /// per operation, per fragment definition). Repeatable directives can be
                    used
                137  +     /// to exhaust CPU and memory resources if not limited.
                138  +     /// </summary>
                139  +     public int MaxAllowedDirectives { get; }
                140  +
                141  +     /// <summary>
                142  +     /// Gets the maximum allowed recursion depth of a parsed document.
```

```
      143  +     /// </summary>
      144  +     public int MaxAllowedRecursionDepth { get; }
      145  +
 89   146        /// <summary>
 90   147        /// Gets the experimental parser options
 91   148        /// which are by default switched of.
```

...f8/Properties/LangUtf8Resources.Designer.cs

Load Diff

Some generated files are not rendered by default. Learn more about customizing how changed files appear on GitHub.

...uage.Utf8/Properties/LangUtf8Resources.resx

```
      @@ -207,4 +207,10 @@
207   207      <data name="Utf8GraphQLParser_Start_MaxAllowedFieldsReached" xml:space="preserve">
208   208        <value>The GraphQL request document contains more than {0} fields. Parsing aborted.</value>
209   209      </data>
      210  +   <data name="Utf8GraphQLParser_ParseDirective_MaxAllowedDirectivesReached" xml:space="preserve">
      211  +     <value>A location in the GraphQL document contains more than {0} directives. Parsing aborted.</value>
      212  +   </data>
      213  +   <data name="Utf8GraphQLParser_Start_MaxAllowedRecursionDepthReached" xml:space="preserve">
      214  +     <value>Document exceeds the maximum allowed recursion depth of {0}. Parsing aborted.</value>
      215  +   </data>
210   216    </root>
```

...nguage.Utf8/Utf8GraphQLParser.Directives.cs

```
      @@ -1,4 +1,5 @@
1     1    using System.Runtime.CompilerServices;
      2  + using static HotChocolate.Language.Properties.LangUtf8Resources;
```

```
2  3
3  4   namespace HotChocolate.Language;
4  5
   @@ -64,7 +65,7 @@ private NameNode ParseDirectiveLocation()
64 65          throw Unexpected(kind);
65 66      }
66 67
67    -     private List<DirectiveNode> ParseDirectives(bool isConstant)
   68 +     private List<DirectiveNode> ParseDirectives(bool isConstant, bool
      isQueryLocation = false)
68 69      {
69 70          if (_reader.Kind == TokenKind.At)
70 71          {
   @@ -73,6 +74,15 @@ private List<DirectiveNode> ParseDirectives(bool
      isConstant)
73 74              while (_reader.Kind == TokenKind.At)
74 75              {
75 76                  list.Add(ParseDirective(isConstant));
   77 +
   78 +                if (isQueryLocation && list.Count > _maxAllowedDirectives)
   79 +                {
   80 +                    throw new SyntaxException(
   81 +                        _reader,
   82 +                        string.Format(
   83 +
      Utf8GraphQLParser_ParseDirective_MaxAllowedDirectivesReached,
   84 +                            _maxAllowedDirectives));
   85 +                }
76 86              }
77 87
78 88              return list;
```

...anguage.Utf8/Utf8GraphQLParser.Fragments.cs

```
   @@ -53,7 +53,7 @@ private FragmentDefinitionNode ParseFragmentDefinition()
53 53                  ParseVariableDefinitions();
54 54              ExpectOnKeyword();
55 55              var typeCondition = ParseNamedType();
56    -             var directives = ParseDirectives(false);
```

```
      56  +             var directives = ParseDirectives(false, isQueryLocation: true);
57    57                var selectionSet = ParseSelectionSet();
58    58                var location = CreateLocation(in start);
59    59
          @@ -72,7 +72,7 @@ private FragmentDefinitionNode ParseFragmentDefinition()
72    72                var name = ParseFragmentName();
73    73                ExpectOnKeyword();
74    74                var typeCondition = ParseNamedType();
75        -             var directives = ParseDirectives(false);
      75  +             var directives = ParseDirectives(false, isQueryLocation: true);
76    76                var selectionSet = ParseSelectionSet();
77    77                var location = CreateLocation(in start);
78    78
          @@ -99,7 +99,7 @@ private FragmentDefinitionNode ParseFragmentDefinition()
99    99            private FragmentSpreadNode ParseFragmentSpread(in TokenInfo start)
100   100           {
101   101               var name = ParseFragmentName();
102       -           var directives = ParseDirectives(false);
      102 +           var directives = ParseDirectives(false, isQueryLocation: true);
103   103               var location = CreateLocation(in start);
104   104
105   105               return new FragmentSpreadNode
          @@ -125,7 +125,7 @@ private InlineFragmentNode ParseInlineFragment(
125   125               in TokenInfo start,
126   126               NamedTypeNode? typeCondition)
127   127           {
128       -           var directives = ParseDirectives(false);
      128 +           var directives = ParseDirectives(false, isQueryLocation: true);
129   129               var selectionSet = ParseSelectionSet();
130   130               var location = CreateLocation(in start);
131   131
```

...nguage.Utf8/Utf8GraphQLParser.Operations.cs

```
          @@ -21,7 +21,7 @@ private OperationDefinitionNode ParseOperationDefinition()
21    21                var operation = ParseOperationType();
22    22                var name = _reader.Kind == TokenKind.Name ? ParseName() : null;
23    23                var variableDefinitions = ParseVariableDefinitions();
24        -             var directives = ParseDirectives(false);
      24  +             var directives = ParseDirectives(false, isQueryLocation: true);
```

```
25  25            var selectionSet = ParseSelectionSet();
26  26            var location = CreateLocation(in start);
27  27
    @@ -127,7 +127,7 @@ private VariableDefinitionNode ParseVariableDefinition()
127 127                ? ParseValueLiteral(true)
128 128                : null;
129 129            var directives =
130     -              ParseDirectives(isConstant: true);
    130 +              ParseDirectives(isConstant: true, isQueryLocation: true);
131 131
132 132            var location = CreateLocation(in start);
133 133
    @@ -163,6 +163,7 @@ private VariableNode ParseVariable()
163 163        /// </summary>
164 164        private SelectionSetNode ParseSelectionSet()
165 165        {
    166 +          IncreaseDepth();
166 167            var start = Start();
167 168
168 169            if (_reader.Kind != TokenKind.LeftBrace)
    @@ -191,6 +192,7 @@ private SelectionSetNode ParseSelectionSet()
191 192
192 193            var location = CreateLocation(in start);
193 194
    195 +          DecreaseDepth();
194 196            return new SelectionSetNode(
195 197                location,
196 198                selections);
    @@ -240,7 +242,7 @@ private FieldNode ParseField()
240 242            }
241 243
242 244            var arguments = ParseArguments(false);
243     -          var directives = ParseDirectives(false);
    245 +          var directives = ParseDirectives(false, isQueryLocation: true);
244 246            var selectionSet = _reader.Kind == TokenKind.LeftBrace
245 247                ? ParseSelectionSet()
```

```
246  248                : null;
```

...rc/Language.Utf8/Utf8GraphQLParser.Types.cs

```
     @@ -12,6 +12,7 @@ public ref partial struct Utf8GraphQLParser
12   12         /// </summary>
13   13         private ITypeNode ParseTypeReference()
14   14         {
     15  +          IncreaseDepth();
15   16             ITypeNode type;
16   17             Location? location;
17   18
     @@ -40,6 +41,7 @@ private ITypeNode ParseTypeReference()
40   41                     MoveNext();
41   42                     location = CreateLocation(in start);
42   43
     44  +                  DecreaseDepth();
43   45                     return new NonNullTypeNode
44   46                     (
45   47                         location,
     @@ -50,6 +52,7 @@ private ITypeNode ParseTypeReference()
50   52                 Unexpected(TokenKind.Bang);
51   53             }
52   54
     55  +          DecreaseDepth();
53   56             return type;
54   57         }
55   58
```

...anguage.Utf8/Utf8GraphQLParser.Utilities.cs

```
     @@ -24,6 +24,25 @@ private NameNode ParseName()
24   24         [MethodImpl(MethodImplOptions.AggressiveInlining)]
25   25         private bool MoveNext() => _reader.MoveNext();
26   26
     27  +      [MethodImpl(MethodImplOptions.AggressiveInlining)]
     28  +      private void IncreaseDepth()
     29  +      {
```

```
      30  +         if (++_recursionDepth > _maxAllowedRecursionDepth)
      31  +         {
      32  +             throw new SyntaxException(
      33  +                 _reader,
      34  +                 string.Format(
      35  +                     Utf8GraphQLParser_Start_MaxAllowedRecursionDepthReached,
      36  +                     _maxAllowedRecursionDepth));
      37  +         }
      38  +     }
      39  +
      40  +     [MethodImpl(MethodImplOptions.AggressiveInlining)]
      41  +     private void DecreaseDepth()
      42  +     {
      43  +         --_recursionDepth;
      44  +     }
      45  +
27    46        [MethodImpl(MethodImplOptions.AggressiveInlining)]
28    47        private TokenInfo Start()
29    48        {
```

...c/Language.Utf8/Utf8GraphQLParser.Values.cs

```
@@ -29,32 +29,37 @@ public ref partial struct Utf8GraphQLParser
29    29        /// </param>
30    30        private IValueNode ParseValueLiteral(bool isConstant)
31    31        {
      32  +         IncreaseDepth();
      33  +
      34  +         IValueNode node;
      35  +
32    36            if (_reader.Kind == TokenKind.LeftBracket)
33    37            {
34        -             return ParseList(isConstant);
      38  +             node = ParseList(isConstant);
35    39            }
36        -
37        -         if (_reader.Kind == TokenKind.LeftBrace)
      40  +         else if (_reader.Kind == TokenKind.LeftBrace)
38    41            {
39        -             return ParseObject(isConstant);
```

```
      42  +              node = ParseObject(isConstant);
40    43                }
41        -
42        -          if (TokenHelper.IsScalarValue(ref _reader))
      44  +          else if (TokenHelper.IsScalarValue(ref _reader))
43    45                {
44        -              return ParseScalarValue();
      46  +              node = ParseScalarValue();
45    47                }
46        -
47        -          if (_reader.Kind == TokenKind.Name)
      48  +          else if (_reader.Kind == TokenKind.Name)
48    49                {
49        -              return ParseEnumValue();
      50  +              node = ParseEnumValue();
50    51                }
51        -
52        -          if (_reader.Kind == TokenKind.Dollar && !isConstant)
      52  +          else if (_reader.Kind == TokenKind.Dollar && !isConstant)
      53  +          {
      54  +              node = ParseVariable();
      55  +          }
      56  +          else
53    57                {
54        -              return ParseVariable();
      58  +              throw Unexpected(_reader.Kind);
55    59                }
56    60
57        -          throw Unexpected(_reader.Kind);
      61  +          DecreaseDepth();
      62  +          return node;
58    63            }
59    64
60    65            [MethodImpl(MethodImplOptions.AggressiveInlining)]
```

...uage/src/Language.Utf8/Utf8GraphQLParser.cs

```
@@ -10,10 +10,13 @@ public ref partial struct Utf8GraphQLParser
10    10            private readonly bool _allowFragmentVars;
11    11            private readonly int _maxAllowedNodes;
```

```
12  12          private readonly int _maxAllowedFields;
    13  +       private readonly int _maxAllowedDirectives;
    14  +       private readonly int _maxAllowedRecursionDepth;
13  15          private Utf8GraphQLReader _reader;
14  16          private StringValueNode? _description;
15  17          private int _parsedNodes;
16  18          private int _parsedFields;
    19  +       private int _recursionDepth;
17  20
18  21          public Utf8GraphQLParser(
19  22              ReadOnlySpan<byte> graphQLData,
        @@ -29,6 +32,8 @@ public Utf8GraphQLParser(
29  32              _allowFragmentVars = options.Experimental.AllowFragmentVariables;
30  33              _maxAllowedNodes = options.MaxAllowedNodes;
31  34              _maxAllowedFields = options.MaxAllowedFields;
    35  +           _maxAllowedDirectives = options.MaxAllowedDirectives;
    36  +           _maxAllowedRecursionDepth = options.MaxAllowedRecursionDepth;
32  37              _reader = new Utf8GraphQLReader(graphQLData, options.MaxAllowedTokens);
33  38              _description = null;
34  39          }
        @@ -47,6 +52,8 @@ internal Utf8GraphQLParser(
47  52              _allowFragmentVars = options.Experimental.AllowFragmentVariables;
48  53              _maxAllowedNodes = options.MaxAllowedNodes;
49  54              _maxAllowedFields = options.MaxAllowedFields;
    55  +           _maxAllowedDirectives = options.MaxAllowedDirectives;
    56  +           _maxAllowedRecursionDepth = options.MaxAllowedRecursionDepth;
50  57              _reader = reader;
51  58              _description = null;
52  59          }
        @@ -64,6 +71,7 @@ internal Utf8GraphQLParser(
64  71          public DocumentNode Parse()
65  72          {
66  73              _parsedNodes = 0;
    74  +           _recursionDepth = 0;
67  75              var definitions = new List<IDefinitionNode>();
68  76
69  77              var start = Start();
```

...t/Language.Tests/Parser/QueryParserTests.cs

```
@@ -6,6 +6,209 @@ namespace HotChocolate.Language;
6  6
7  7   public class QueryParserTests
8  8   {
9  +       [Fact]
10 +       public void Default_MaxAllowedRecursionDepth_Is_200()
11 +       {
12 +           Assert.Equal(200, ParserOptions.Default.MaxAllowedRecursionDepth);
13 +       }
14 +
15 +       [Fact]
16 +       public void Reject_Queries_Exceeding_Max_Recursion_Depth_Selection_Sets()
17 +       {
18 +           const int depth = 201;
19 +           var query = string.Concat(Enumerable.Repeat("{ a", depth))
20 +               + string.Concat(Enumerable.Repeat(" }", depth));
21 +
22 +           Assert
23 +               .Throws<SyntaxException>(() => Utf8GraphQLParser.Parse(query))
24 +               .Message
25 +               .MatchInlineSnapshot(
26 +                   "Document exceeds the maximum allowed recursion depth of 200.
   Parsing aborted.");
27 +       }
28 +
29 +       [Fact]
30 +       public void Reject_Queries_Exceeding_Max_Recursion_Depth_Object_Values()
31 +       {
32 +           const int depth = 201;
33 +           var query = "{ a(x: "
34 +               + string.Concat(Enumerable.Repeat("{a: ", depth))
35 +               + "1"
36 +               + string.Concat(Enumerable.Repeat("}", depth))
37 +               + ") }";
38 +
39 +           Assert
40 +               .Throws<SyntaxException>(() => Utf8GraphQLParser.Parse(query))
41 +               .Message
42 +               .MatchInlineSnapshot(
```

```
43  +                 "Document exceeds the maximum allowed recursion depth of 200.
    Parsing aborted.");
44  +     }
45  +
46  +     [Fact]
47  +     public void Reject_Queries_Exceeding_Max_Recursion_Depth_List_Values()
48  +     {
49  +         const int depth = 201;
50  +         var query = "{ a(x: "
51  +             + string.Concat(Enumerable.Repeat("[", depth))
52  +             + "1"
53  +             + string.Concat(Enumerable.Repeat("]", depth))
54  +             + ") }";
55  +
56  +         Assert
57  +             .Throws<SyntaxException>(() => Utf8GraphQLParser.Parse(query))
58  +             .Message
59  +             .MatchInlineSnapshot(
60  +                 "Document exceeds the maximum allowed recursion depth of 200.
    Parsing aborted.");
61  +     }
62  +
63  +     [Fact]
64  +     public void Reject_Queries_Exceeding_Max_Recursion_Depth_List_Types()
65  +     {
66  +         const int depth = 201;
67  +         var query = $"query($v: {string.Concat(Enumerable.Repeat("[",
    depth))}Int{string.Concat(Enumerable.Repeat("]", depth))}) {{ a }}";
68  +
69  +         Assert
70  +             .Throws<SyntaxException>(() => Utf8GraphQLParser.Parse(query))
71  +             .Message
72  +             .MatchInlineSnapshot(
73  +                 "Document exceeds the maximum allowed recursion depth of 200.
    Parsing aborted.");
74  +     }
75  +
76  +     [Fact]
77  +     public void Allow_Queries_Within_Max_Recursion_Depth()
78  +     {
```

```
79  +         const int depth = 50;
80  +         var query = string.Concat(Enumerable.Repeat("{ a", depth))
81  +             + string.Concat(Enumerable.Repeat(" }", depth));
82  +
83  +         var document = Utf8GraphQLParser.Parse(query);
84  +
85  +         Assert.NotNull(document);
86  +         Assert.Single(document.Definitions);
87  +     }
88  +
89  +     [Fact]
90  +     public void Reject_Queries_Exceeding_Custom_Recursion_Depth()
91  +     {
92  +         var options = new ParserOptions(
93  +             noLocations: false,
94  +             allowFragmentVariables: false,
95  +             maxAllowedNodes: int.MaxValue,
96  +             maxAllowedTokens: int.MaxValue,
97  +             maxAllowedFields: 2048,
98  +             maxAllowedDirectives: 4,
99  +             maxAllowedRecursionDepth: 10);
100 +         const int depth = 11;
101 +         var query = string.Concat(Enumerable.Repeat("{ a", depth))
102 +             + string.Concat(Enumerable.Repeat(" }", depth));
103 +
104 +         Assert
105 +             .Throws<SyntaxException>(() => Utf8GraphQLParser.Parse(query,
    options))
106 +             .Message
107 +             .MatchInlineSnapshot(
108 +                 "Document exceeds the maximum allowed recursion depth of 10.
    Parsing aborted.");
109 +     }
110 +
111 +     [Fact]
112 +     public void Allow_Queries_Within_Custom_Recursion_Depth()
113 +     {
114 +         var options = new ParserOptions(
115 +             noLocations: false,
116 +             allowFragmentVariables: false,
```

```
117 +                maxAllowedNodes: int.MaxValue,
118 +                maxAllowedTokens: int.MaxValue,
119 +                maxAllowedFields: 2048,
120 +                maxAllowedDirectives: 4,
121 +                maxAllowedRecursionDepth: 10);
122 +            const int depth = 10;
123 +            var query = string.Concat(Enumerable.Repeat("{ a", depth))
124 +                + string.Concat(Enumerable.Repeat(" }", depth));
125 +
126 +            var document = Utf8GraphQLParser.Parse(query, options);
127 +
128 +            Assert.NotNull(document);
129 +            Assert.Single(document.Definitions);
130 +        }
131 +
132 +        [Theory]
133 +        [InlineData(20_000)]
134 +        [InlineData(50_000)]
135 +        public void Reject_Attack_Payload_Nested_Selection_Sets(int depth)
136 +        {
137 +            var query = string.Concat(Enumerable.Repeat("{ a", depth))
138 +                + string.Concat(Enumerable.Repeat(" }", depth));
139 +
140 +            Assert.Throws<SyntaxException>(() => Utf8GraphQLParser.Parse(query));
141 +        }
142 +
143 +        [Theory]
144 +        [InlineData(20_000)]
145 +        [InlineData(50_000)]
146 +        public void Reject_Attack_Payload_Nested_List_Values(int depth)
147 +        {
148 +            var query = "{ a(x: "
149 +                + string.Concat(Enumerable.Repeat("[", depth))
150 +                + "1"
151 +                + string.Concat(Enumerable.Repeat("]", depth))
152 +                + ") }";
153 +
154 +            Assert.Throws<SyntaxException>(() => Utf8GraphQLParser.Parse(query));
155 +        }
156 +
```

```
157 +     [Fact]
158 +     public void Default_MaxAllowedDirectives_Is_4()
159 +     {
160 +         Assert.Equal(4, ParserOptions.Default.MaxAllowedDirectives);
161 +     }
162 +
163 +     [Fact]
164 +     public void Reject_Fields_Exceeding_Max_Allowed_Directives_Per_Location()
165 +     {
166 +         Assert
167 +             .Throws<SyntaxException>(() => Utf8GraphQLParser.Parse("{ a @d @d
      @d @d @d }"))
168 +             .Message
169 +             .MatchInlineSnapshot(
170 +                 "A location in the GraphQL document contains more than 4
      directives. Parsing aborted.");
171 +     }
172 +
173 +     [Fact]
174 +     public void Allow_Fields_Within_Max_Allowed_Directives_Per_Location()
175 +     {
176 +         Utf8GraphQLParser.Parse("{ a @d @d @d @d }");
177 +     }
178 +
179 +     [Fact]
180 +     public void Reject_Fields_Exceeding_Custom_Directive_Limit()
181 +     {
182 +         var options = new ParserOptions(
183 +             noLocations: false,
184 +             allowFragmentVariables: false,
185 +             maxAllowedNodes: int.MaxValue,
186 +             maxAllowedTokens: int.MaxValue,
187 +             maxAllowedFields: 2048,
188 +             maxAllowedDirectives: 2,
189 +             maxAllowedRecursionDepth: 200);
190 +
191 +         Assert
192 +             .Throws<SyntaxException>(() => Utf8GraphQLParser.Parse("{ a @d @d
      @d }", options))
193 +             .Message
```

```
194 +             .MatchInlineSnapshot(
195 +                 "A location in the GraphQL document contains more than 2
      directives. Parsing aborted.");
196 +     }
197 +
198 +     [Fact]
199 +     public void Allow_Fields_Within_Custom_Directive_Limit()
200 +     {
201 +         var options = new ParserOptions(
202 +             noLocations: false,
203 +             allowFragmentVariables: false,
204 +             maxAllowedNodes: int.MaxValue,
205 +             maxAllowedTokens: int.MaxValue,
206 +             maxAllowedFields: 2048,
207 +             maxAllowedDirectives: 2,
208 +             maxAllowedRecursionDepth: 200);
209 +         Utf8GraphQLParser.Parse("{ a @d @d }", options);
210 +     }
211 +
9   212       [Fact]
10  213       public void Reject_Queries_With_More_Than_2048_Fields()
11  214       {;
```

Comments 0

Please sign in to comment.