

LinksPlatform's Platform.Communication.Protocol.Lino Class Library

1.1 ./csharp/Platform.Communication.Protocol.Lino/ILinksGroupListExtensions.cs

```
1 using System.Collections.Generic;
2 using System.Runtime.CompilerServices;
3
4 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
6 namespace Platform.Communication.Protocol.Lino
7 {
8     /// <summary>
9     /// <para>
10    /// Represents the links group list extensions.
11    /// </para>
12    /// <para></para>
13    /// </summary>
14    public static class ILinksGroupListExtensions
15    {
16        /// <summary>
17        /// <para>
18        /// Returns the links list using the specified groups.
19        /// </para>
20        /// <para></para>
21        /// </summary>
22        /// <param name="groups">
23        /// <para>The groups.</para>
24        /// <para></para>
25        /// </param>
26        /// <returns>
27        /// <para>The list.</para>
28        /// <para></para>
29        /// </returns>
30        [MethodImpl(MethodImplOptions.AggressiveInlining)]
31        public static List<Link> ToLinksList(this IList<LinksGroup> groups)
32        {
33            var list = new List<Link>();
34            for (var i = 0; i < groups.Count; i++)
35            {
36                groups[i].AppendToLinksList(list);
37            }
38            return list;
39        }
40    }
41 }
```

1.2 ./csharp/Platform.Communication.Protocol.Lino/IListExtensions.cs

```
1 using Platform.Collections;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Runtime.CompilerServices;
6
7 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
8
9 namespace Platform.Communication.Protocol.Lino
10 {
11     /// <summary>
12     /// <para>
13     /// Represents the list extensions.
14     /// </para>
15     /// <para></para>
16     /// </summary>
17     public static class IListExtensions
18     {
19         /// <summary>
20         /// <para>
21         /// Formats the links.
22         /// </para>
23         /// <para></para>
24         /// </summary>
25         /// <param name="links">
26         /// <para>The links.</para>
27         /// <para></para>
28         /// </param>
29         /// <returns>
30         /// <para>The string</para>
31         /// <para></para>
32         /// </returns>
33         [MethodImpl(MethodImplOptions.AggressiveInlining)]
```

```

34     public static string Format(this IList<Link> links) => string.Join(Environment.NewLine,
35         ↪ links.Select(l => l.ToString()));
36
37     /// <summary>
38     /// <para>
39     /// Formats the links.
40     /// </para>
41     /// <para></para>
42     /// </summary>
43     /// <param name="links">
44     /// <para>The links.</para>
45     /// <para></para>
46     /// </param>
47     /// <param name="lessParentheses">
48     /// <para>The less parentheses.</para>
49     /// <para></para>
50     /// </param>
51     /// <returns>
52     /// <para>The string</para>
53     /// <para></para>
54     /// </returns>
55     [MethodImpl(MethodImplOptions.AggressiveInlining)]
56     public static string Format(this IList<Link> links, bool lessParentheses)
57     {
58         if (lessParentheses == false)
59         {
60             return links.Format();
61         }
62         else
63         {
64             return string.Join(Environment.NewLine, links.Select(l =>
65                 ↪ l.ToString().TrimSingle('(').TrimSingle(')')));
66         }
67     }
68 }

```

1.3 ./csharp/Platform.Communication.Protocol.Lino/Link.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Runtime.CompilerServices;
4  using System.Text;
5  using Platform.Collections;
6  using Platform.Collections.Lists;
7
8  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
9
10 namespace Platform.Communication.Protocol.Lino
11 {
12     /// <summary>
13     /// <para>
14     /// The link.
15     /// </para>
16     /// <para></para>
17     /// </summary>
18     public struct Link : IEquatable<Link>
19     {
20         /// <summary>
21         /// <para>
22         /// The id.
23         /// </para>
24         /// <para></para>
25         /// </summary>
26         public readonly string Id;
27
28         /// <summary>
29         /// <para>
30         /// The values.
31         /// </para>
32         /// <para></para>
33         /// </summary>
34         public readonly IList<Link> Values;
35
36         /// <summary>
37         /// <para>
38         /// Initializes a new <see cref="Link"/> instance.
39         /// </para>
40         /// <para></para>
41         /// </summary>

```

```

42    /// <param name="id">
43    /// <para>A id.</para>
44    /// <para></para>
45    /// </param>
46    /// <param name="values">
47    /// <para>A values.</para>
48    /// <para></para>
49    /// </param>
50    [MethodImpl(MethodImplOptions.AggressiveInlining)]
51    public Link(string id, IList<Link> values) => (Id, Values) = (id, values);
52
53    /// <summary>
54    /// <para>
55    /// Initializes a new <see cref="Link"/> instance.
56    /// </para>
57    /// <para></para>
58    /// </summary>
59    /// <param name="values">
60    /// <para>A values.</para>
61    /// <para></para>
62    /// </param>
63    [MethodImpl(MethodImplOptions.AggressiveInlining)]
64    public Link(IList<Link> values) : this(null, values) { }
65
66    /// <summary>
67    /// <para>
68    /// Initializes a new <see cref="Link"/> instance.
69    /// </para>
70    /// <para></para>
71    /// </summary>
72    /// <param name="values">
73    /// <para>A values.</para>
74    /// <para></para>
75    /// </param>
76    [MethodImpl(MethodImplOptions.AggressiveInlining)]
77    public Link(params Link[] values) : this(null, values) { }
78
79    /// <summary>
80    /// <para>
81    /// Initializes a new <see cref="Link"/> instance.
82    /// </para>
83    /// <para></para>
84    /// </summary>
85    /// <param name="id">
86    /// <para>A id.</para>
87    /// <para></para>
88    /// </param>
89    [MethodImpl(MethodImplOptions.AggressiveInlining)]
90    public Link(string id) : this(id, null) { }
91
92    /// <summary>
93    /// <para>
94    /// Returns the string.
95    /// </para>
96    /// <para></para>
97    /// </summary>
98    /// <returns>
99    /// <para>The string</para>
100    /// <para></para>
101    /// </returns>
102    [MethodImpl(MethodImplOptions.AggressiveInlining)]
103    public override string ToString() => Values.IsNullOrEmpty() ? $"{(Id)}" :
    ↪ GetLinkValuesString();
104
105    /// <summary>
106    /// <para>
107    /// Gets the link values string.
108    /// </para>
109    /// <para></para>
110    /// </summary>
111    /// <returns>
112    /// <para>The string</para>
113    /// <para></para>
114    /// </returns>
115    [MethodImpl(MethodImplOptions.AggressiveInlining)]
116    private string GetLinkValuesString() => Id == null ? $"{(GetValuesString())}" :
    ↪ $"{(Id): {GetValuesString()}}";
117

```

```

118     /// <summary>
119     /// <para>
120     /// Gets the values string.
121     /// </para>
122     /// <para></para>
123     /// </summary>
124     /// <returns>
125     /// <para>The string</para>
126     /// <para></para>
127     /// </returns>
128     [MethodImpl(MethodImplOptions.AggressiveInlining)]
129     public string GetValuesString()
130     {
131         var sb = new StringBuilder();
132         for (int i = 0; i < Values.Count; i++)
133         {
134             if (i > 0)
135             {
136                 sb.Append(' ');
137             }
138             sb.Append(GetValueString(Values[i]));
139         }
140         return sb.ToString();
141     }
142
143     /// <summary>
144     /// <para>
145     /// Simplifies this instance.
146     /// </para>
147     /// <para></para>
148     /// </summary>
149     /// <returns>
150     /// <para>The link</para>
151     /// <para></para>
152     /// </returns>
153     [MethodImpl(MethodImplOptions.AggressiveInlining)]
154     public Link Simplify()
155     {
156         if (Values.IsNullOrEmpty())
157         {
158             return this;
159         }
160         else if (Values.Count == 1)
161         {
162             return Values[0];
163         }
164         else
165         {
166             var newValues = new Link[Values.Count];
167             for (int i = 0; i < Values.Count; i++)
168             {
169                 newValues[i] = Values[i].Simplify();
170             }
171             return new Link(Id, newValues);
172         }
173     }
174
175     /// <summary>
176     /// <para>
177     /// Combines the other.
178     /// </para>
179     /// <para></para>
180     /// </summary>
181     /// <param name="other">
182     /// <para>The other.</para>
183     /// <para></para>
184     /// </param>
185     /// <returns>
186     /// <para>The link</para>
187     /// <para></para>
188     /// </returns>
189     [MethodImpl(MethodImplOptions.AggressiveInlining)]
190     public Link Combine(Link other) => new Link(this, other);
191
192     /// <summary>
193     /// <para>
194     /// Gets the value string using the specified value.
195     /// </para>

```

```

196     /// <para></para>
197     /// </summary>
198     /// <param name="value">
199     /// <para>The value.</para>
200     /// <para></para>
201     /// </param>
202     /// <returns>
203     /// <para>The string</para>
204     /// <para></para>
205     /// </returns>
206     [MethodImpl(MethodImplOptions.AggressiveInlining)]
207     public static string GetValueString(Link value) => value.ToLinkOrIdString();
208
209     /// <summary>
210     /// <para>
211     /// Returns the link or id string.
212     /// </para>
213     /// <para></para>
214     /// </summary>
215     /// <returns>
216     /// <para>The string</para>
217     /// <para></para>
218     /// </returns>
219     [MethodImpl(MethodImplOptions.AggressiveInlining)]
220     public string ToLinkOrIdString() => Values.IsNullOrEmpty() ? Id : ToString();
221
222     [MethodImpl(MethodImplOptions.AggressiveInlining)]
223     public static implicit operator Link(string value) => new Link(value);
224
225     [MethodImpl(MethodImplOptions.AggressiveInlining)]
226     public static implicit operator Link((string, IList<Link>) value) => new
227     ↪ Link(value.Item1, value.Item2);
228
229     [MethodImpl(MethodImplOptions.AggressiveInlining)]
230     public static implicit operator Link((Link, Link) value) => new Link(value.Item1,
231     ↪ value.Item2);
232
233     [MethodImpl(MethodImplOptions.AggressiveInlining)]
234     public static implicit operator Link((Link, Link, Link) value) => new Link(value.Item1,
235     ↪ value.Item2, value.Item3);
236
237     /// <summary>
238     /// <para>
239     /// Determines whether this instance equals.
240     /// </para>
241     /// <para></para>
242     /// </summary>
243     /// <param name="obj">
244     /// <para>The obj.</para>
245     /// <para></para>
246     /// </param>
247     /// <returns>
248     /// <para>The bool</para>
249     /// <para></para>
250     /// </returns>
251     [MethodImpl(MethodImplOptions.AggressiveInlining)]
252     public override bool Equals(object obj) => obj is Link link ? Equals(link) : false;
253
254     /// <summary>
255     /// <para>
256     /// Gets the hash code.
257     /// </para>
258     /// <para></para>
259     /// </summary>
260     /// <returns>
261     /// <para>The int</para>
262     /// <para></para>
263     /// </returns>
264     [MethodImpl(MethodImplOptions.AggressiveInlining)]
265     public override int GetHashCode() => (Id, Values.GenerateHashCode()).GetHashCode();
266
267     /// <summary>
268     /// <para>
269     /// Determines whether this instance equals.
270     /// </para>
271     /// <para></para>
272     /// </summary>
273     /// <param name="other">

```

```

271     /// <para>The other.</para>
272     /// <para></para>
273     /// </param>
274     /// <returns>
275     /// <para>The bool</para>
276     /// <para></para>
277     /// </returns>
278     [MethodImpl(MethodImplOptions.AggressiveInlining)]
279     public bool Equals(Link other) => Id == other.Id && Values.EqualTo(other.Values);
280
281     [MethodImpl(MethodImplOptions.AggressiveInlining)]
282     public static bool operator ==(Link left, Link right) => left.Equals(right);
283
284     [MethodImpl(MethodImplOptions.AggressiveInlining)]
285     public static bool operator !=(Link left, Link right) => !(left == right);
286 }
287 }

```

1.4 ./csharp/Platform.Communication.Protocol.Lino/LinksGroup.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Runtime.CompilerServices;
4  using Platform.Collections;
5  using Platform.Collections.Lists;
6
7  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
8
9  namespace Platform.Communication.Protocol.Lino
10 {
11     /// <summary>
12     /// <para>
13     /// The links group.
14     /// </para>
15     /// <para></para>
16     /// </summary>
17     public struct LinksGroup : IEquatable<LinksGroup>
18     {
19         /// <summary>
20         /// <para>
21         /// Gets or sets the link value.
22         /// </para>
23         /// <para></para>
24         /// </summary>
25         public Link Link
26         {
27             [MethodImpl(MethodImplOptions.AggressiveInlining)]
28             get;
29             [MethodImpl(MethodImplOptions.AggressiveInlining)]
30             set;
31         }
32
33         /// <summary>
34         /// <para>
35         /// Gets or sets the groups value.
36         /// </para>
37         /// <para></para>
38         /// </summary>
39         public IList<LinksGroup> Groups
40         {
41             [MethodImpl(MethodImplOptions.AggressiveInlining)]
42             get;
43             [MethodImpl(MethodImplOptions.AggressiveInlining)]
44             set;
45         }
46
47         /// <summary>
48         /// <para>
49         /// Initializes a new <see cref="LinksGroup"/> instance.
50         /// </para>
51         /// <para></para>
52         /// </summary>
53         /// <param name="link">
54         /// <para>A link.</para>
55         /// <para></para>
56         /// </param>
57         /// <param name="groups">
58         /// <para>A groups.</para>
59         /// <para></para>
60         /// </param>

```

```

61 [MethodImpl(MethodImplOptions.AggressiveInlining)]
62 public LinksGroup(Link link, IList<LinksGroup> groups)
63 {
64     Link = link;
65     Groups = groups;
66 }
67
68 /// <summary>
69 /// <para>
70 /// Initializes a new <see cref="LinksGroup"/> instance.
71 /// </para>
72 /// <para></para>
73 /// </summary>
74 /// <param name="link">
75 /// <para>A link.</para>
76 /// <para></para>
77 /// </param>
78 [MethodImpl(MethodImplOptions.AggressiveInlining)]
79 public LinksGroup(Link link) : this(link, null) { }
80
81 [MethodImpl(MethodImplOptions.AggressiveInlining)]
82 public static implicit operator List<Link>(LinksGroup value) => value.ToLinksList();
83
84 /// <summary>
85 /// <para>
86 /// Returns the links list.
87 /// </para>
88 /// <para></para>
89 /// </summary>
90 /// <returns>
91 /// <para>The list.</para>
92 /// <para></para>
93 /// </returns>
94 [MethodImpl(MethodImplOptions.AggressiveInlining)]
95 public List<Link> ToLinksList()
96 {
97     var list = new List<Link>();
98     AppendToLinksList(list);
99     return list;
100 }
101
102 /// <summary>
103 /// <para>
104 /// Appends the to links list using the specified list.
105 /// </para>
106 /// <para></para>
107 /// </summary>
108 /// <param name="list">
109 /// <para>The list.</para>
110 /// <para></para>
111 /// </param>
112 [MethodImpl(MethodImplOptions.AggressiveInlining)]
113 public void AppendToLinksList(List<Link> list) => AppendToLinksList(list, Link, this);
114
115 /// <summary>
116 /// <para>
117 /// Appends the to links list using the specified list.
118 /// </para>
119 /// <para></para>
120 /// </summary>
121 /// <param name="list">
122 /// <para>The list.</para>
123 /// <para></para>
124 /// </param>
125 /// <param name="dependency">
126 /// <para>The dependency.</para>
127 /// <para></para>
128 /// </param>
129 /// <param name="group">
130 /// <para>The group.</para>
131 /// <para></para>
132 /// </param>
133 [MethodImpl(MethodImplOptions.AggressiveInlining)]
134 public static void AppendToLinksList(List<Link> list, Link dependency, LinksGroup group)
135 {
136     list.Add(dependency);
137     var groups = group.Groups;
138     if (!groups.IsNullOrEmpty())

```

```

139     {
140         for (int i = 0; i < groups.Count; i++)
141         {
142             var innerGroup = groups[i];
143             AppendToLinksList(list, dependency.Combine(innerGroup.Link), innerGroup);
144         }
145     }
146 }
147
148 /// <summary>
149 /// <para>
150 /// Determines whether this instance equals.
151 /// </para>
152 /// <para></para>
153 /// </summary>
154 /// <param name="obj">
155 /// <para>The obj.</para>
156 /// <para></para>
157 /// </param>
158 /// <returns>
159 /// <para>The bool</para>
160 /// <para></para>
161 /// </returns>
162 [MethodImpl(MethodImplOptions.AggressiveInlining)]
163 public override bool Equals(object obj) => obj is LinksGroup linksGroup ?
    ↳ Equals(linksGroup) : false;
164
165 /// <summary>
166 /// <para>
167 /// Gets the hash code.
168 /// </para>
169 /// <para></para>
170 /// </summary>
171 /// <returns>
172 /// <para>The int</para>
173 /// <para></para>
174 /// </returns>
175 [MethodImpl(MethodImplOptions.AggressiveInlining)]
176 public override int GetHashCode() => (Link, Groups.GenerateHashCode()).GetHashCode();
177
178 /// <summary>
179 /// <para>
180 /// Determines whether this instance equals.
181 /// </para>
182 /// <para></para>
183 /// </summary>
184 /// <param name="other">
185 /// <para>The other.</para>
186 /// <para></para>
187 /// </param>
188 /// <returns>
189 /// <para>The bool</para>
190 /// <para></para>
191 /// </returns>
192 [MethodImpl(MethodImplOptions.AggressiveInlining)]
193 public bool Equals(LinksGroup other) => Link == other.Link &&
    ↳ Groups.EqualTo(other.Groups);
194
195 [MethodImpl(MethodImplOptions.AggressiveInlining)]
196 public static bool operator ==(LinksGroup left, LinksGroup right) => left.Equals(right);
197
198 [MethodImpl(MethodImplOptions.AggressiveInlining)]
199 public static bool operator !=(LinksGroup left, LinksGroup right) => !(left == right);
200 }
201 }

```

1.5 ./csharp/Platform.Communication.Protocol.Lino/_cs

```

1 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
2
3 using System.Runtime.CompilerServices;
4
5 namespace Platform.Communication.Protocol.Lino
6 {
7     /// <summary>
8     /// <para>
9     /// The .
10    /// </para>
11    /// <para></para>

```



```

12     /// </summary>
13     public struct _
14     {
15         /// <summary>
16         /// <para>
17         /// The link.
18         /// </para>
19         /// <para></para>
20         /// </summary>
21         public readonly Link Link;
22
23         /// <summary>
24         /// <para>
25         /// Initializes a new <see cref="_"/> instance.
26         /// </para>
27         /// <para></para>
28         /// </summary>
29         /// <param name="id">
30         /// <para>A id.</para>
31         /// <para></para>
32         /// </param>
33         [MethodImpl(MethodImplOptions.AggressiveInlining)]
34         public _(Link id) => Link = id;
35
36         [MethodImpl(MethodImplOptions.AggressiveInlining)]
37         public static implicit operator _(Link value) => new _(value);
38
39         [MethodImpl(MethodImplOptions.AggressiveInlining)]
40         public static implicit operator _(string id) => new _(id);
41
42         [MethodImpl(MethodImplOptions.AggressiveInlining)]
43         public static implicit operator _((string, Link) value) => new Link(value.Item1, new
44             ↳ Link[] { value.Item2 });
45
46         [MethodImpl(MethodImplOptions.AggressiveInlining)]
47         public static implicit operator _((string, Link, Link) value) => new Link(value.Item1,
48             ↳ new Link[] { value.Item2, value.Item3 });
49
50         [MethodImpl(MethodImplOptions.AggressiveInlining)]
51         public static implicit operator _((string, Link, Link, Link) value) => new
52             ↳ Link(value.Item1, new Link[] { value.Item2, value.Item3, value.Item4 });
53
54         [MethodImpl(MethodImplOptions.AggressiveInlining)]
55         public static implicit operator Link(_ value) => value.Link;
56     }
57 }

```

1.6 ./csharp/Platform.Communication.Protocol.Lino.Tests/ParserTests.cs

```

1     using Xunit;
2
3     namespace Platform.Communication.Protocol.Lino.Tests
4     {
5         /// <summary>
6         /// <para>
7         /// Represents the parser tests.
8         /// </para>
9         /// <para></para>
10        /// </summary>
11        public static class ParserTests
12        {
13            /// <summary>
14            /// <para>
15            /// Tests that single link test.
16            /// </para>
17            /// <para></para>
18            /// </summary>
19            [Fact]
20            public static void SingleLinkTest()
21            {
22                var source = @"(address: source target)";
23                var parser = new Parser();
24                var links = parser.Parse(source);
25                var target = links.Format();
26                Assert.Equal(source, target);
27            }
28
29            /// <summary>
30            /// <para>
31            /// Tests that two links test.

```

```

32     /// </para>
33     /// <para></para>
34     /// </summary>
35     [Fact]
36     public static void TwoLinksTest()
37     {
38         var source = @"(first: x y
39 (second: a b)";
40         var parser = new Parser();
41         var links = parser.Parse(source);
42         var target = links.Format();
43         Assert.Equal(source, target);
44     }
45
46     /// <summary>
47     /// <para>
48     /// Tests that parse and stringify test.
49     /// </para>
50     /// <para></para>
51     /// </summary>
52     [Fact]
53     public static void ParseAndStringifyTest()
54     {
55         var source = @"(papa (lovesMama: loves mama))
56 (son lovesMama)
57 (daughter lovesMama)
58 (all (love mama))";
59         var parser = new Parser();
60         var links = parser.Parse(source);
61         var target = links.Format();
62         Assert.Equal(source, target);
63     }
64
65     /// <summary>
66     /// <para>
67     /// Tests that bug test.
68     /// </para>
69     /// <para></para>
70     /// </summary>
71     [Fact]
72     public static void BugTest()
73     {
74         var source = @"(ignore conan-center-index repository)";
75         var links = (new Platform.Communication.Protocol.Lino.Parser()).Parse(source);
76         var target = links.Format();
77         Assert.Equal(source, target);
78     }
79
80     /// <summary>
81     /// <para>
82     /// Tests that parse and stringify with less parentheses test.
83     /// </para>
84     /// <para></para>
85     /// </summary>
86     [Fact]
87     public static void ParseAndStringifyWithLessParenthesesTest()
88     {
89         var source = @"lovesMama: loves mama
90 papa lovesMama
91 son lovesMama
92 daughter lovesMama
93 all (love mama)";
94         var parser = new Parser();
95         var links = parser.Parse(source);
96         var target = links.Format(lessParentheses: true);
97         Assert.Equal(source, target);
98     }
99
100    /// <summary>
101    /// <para>
102    /// Tests that significant whitespace test.
103    /// </para>
104    /// <para></para>
105    /// </summary>
106    [Fact]
107    public static void SignificantWhitespaceTest()
108    {
109        var source = @"

```

```

110 users
111     user1
112         id
113             43
114         name
115             first
116                 John
117             last
118                 Williams
119         location
120             New York
121         age
122             23
123     user2
124         id
125             56
126         name
127             first
128                 Igor
129             middle
130                 Petrovich
131             last
132                 Ivanov
133         location
134             Moscow
135         age
136             20";
137     var target = @"(users)
138 (users user1)
139 ((users user1) id)
140 (((users user1) id) 43)
141 ((users user1) name)
142 (((users user1) name) first)
143 (((users user1) name) first) John)
144 ((users user1) name) last)
145 (((users user1) name) last) Williams)
146 (users user1) location)
147 (((users user1) location) (New York))
148 (users user1) age)
149 (((users user1) age) 23)
150 (users user2)
151 ((users user2) id)
152 (((users user2) id) 56)
153 ((users user2) name)
154 (((users user2) name) first)
155 (((users user2) name) first) Igor)
156 ((users user2) name) middle)
157 (((users user2) name) middle) Petrovich)
158 ((users user2) name) last)
159 (((users user2) name) last) Ivanov)
160 (users user2) location)
161 (((users user2) location) Moscow)
162 (users user2) age)
163 (((users user2) age) 20)";
164     var parser = new Parser();
165     var links = parser.Parse(source);
166     var formattedLinks = links.Format();
167     Assert.Equal(target, formattedLinks);
168 }
169 }
170 }

```

1.7 ./csharp/Platform.Communication.Protocol.Lino.Tests/TupleTests.cs

```

1 using System.Collections.Generic;
2 using Xunit;
3
4 namespace Platform.Communication.Protocol.Lino.Tests
5 {
6     /// <summary>
7     /// <para>
8     /// Represents the tuple tests.
9     /// </para>
10    /// <para></para>
11    /// </summary>
12    public class TupleTests
13    {
14        /// <summary>
15        /// <para>
16        /// Tests that tuple to link test.
17        /// </para>

```

```

18     /// <para></para>
19     /// </summary>
20     [Fact]
21     public void TupleToLinkTest()
22     {
23         var source = @"(papa (lovesMama: loves mama))
24 (son lovesMama)
25 (daughter lovesMama)
26 (all (love mama))";
27         var parser = new Parser();
28         var links = parser.Parse(source);
29         var targetFromString = links.Format();
30
31         IList<Link> constructedLinks = new List<Link>()
32         {
33             ("papa", (_)("lovesMama", "loves", "mama")),
34             ("son", "lovesMama"),
35             ("daughter", "lovesMama"),
36             ("all", ("love", "mama")),
37         };
38         var targetFromTuples = constructedLinks.Format();
39         Assert.Equal(targetFromString, targetFromTuples);
40     }
41 }
42 }

```

Index

./csharp/Platform.Communication.Protocol.Lino.Tests/ParserTests.cs, 9
./csharp/Platform.Communication.Protocol.Lino.Tests/TupleTests.cs, 11
./csharp/Platform.Communication.Protocol.Lino/ILinksGroupListExtensions.cs, 1
./csharp/Platform.Communication.Protocol.Lino/IListExtensions.cs, 1
./csharp/Platform.Communication.Protocol.Lino/Link.cs, 2
./csharp/Platform.Communication.Protocol.Lino/LinksGroup.cs, 6
./csharp/Platform.Communication.Protocol.Lino/_.cs, 8