# A Linear Decomposition of Multiparty Sessions for Safe Distributed Programming (Artifact)*

**Alceste Scalas[1], Ornela Dardha[2], Raymond Hu[3], and Nobuko Yoshida[4]**

1 **Imperial College London, UK**
`alceste.scalas@imperial.ac.uk`
2 **University of Glasgow, UK**
`ornela.dardha@glasgow.ac.uk`
3 **Imperial College London, UK**
`raymond.hu@imperial.ac.uk`
4 **Imperial College London, UK**
`n.yoshida@imperial.ac.uk`

## Abstract

This artifact contains a version of the Scribble tool that, given a protocol specification with multiple participants, can generate Scala APIs for implementing each participant in a type-safe, protocol-abiding way. Crucially, the API generation leverages a *decomposition* of the multiparty protocol into type-safe peer-to-peer interactions between pairs of participants; and this, in turn, allows to implement the API internals on top of the existing `lchannels` library for type-safe *binary* session programming. As a result, several technically challenging aspects in the implementation of multiparty sessions are solved "for free", at the underlying binary level. This includes *distributed multiparty session delegation*: this artifact implements it for the first time.

**1998 ACM Subject Classification** D.1.3 Concurrent Programming; D.3.1 Formal Definitions and Theory; F.3.3 Studies of Program Constructs — Type structure

**Keywords and phrases** process calculi, session types, concurrent programming, Scala

**Digital Object Identifier** 10.4230/DARTS.3.2.1

**Related Conference** European Conference on Object-Oriented Programming (ECOOP 2017), June 18-23, 2017, Barcelona, Spain

## 1 Scope

This artifact presents an application of the formal *multiparty-to-binary session decomposition* studied in the companion paper.

This artifact shows that the theoretical results of the companion paper provide the basis for automatically generating Scala APIs for type-safe distributed programming. Moreover, the artifact shows that such a theoretically-grounded approach brings a relevant technical simplification over previous implementations of multiparty sessions: the generated APIs contain very little logic, and are just a thin layer on top of the existing `lchannels` library [2, 3], that handles many irksome issues. This simplification yields, in particular, the first implementation of *distributed multiparty delegation*.

Technically, the API generation has been implemented by extending the Scribble tool [1].

* This artifact is a companion of the paper: A. Scalas, O. Dardha, R. Hu, N. Yoshida, "A Linear Decomposition of Multiparty Sessions for Safe Distributed Programming", Proceedings of the 31st European Conference on Object-Oriented Programming (ECOOP 2017), June 18-23, 2017, Barcelona, Spain. This work was supported in part by EPSRC (grants EP/K034413/1, EP/K011715/1, EP/L00058X/1, EP/N027833/1, EP/N028201/1) and EU (FP7 612985 "Upscale"). Dardha was awarded a Postdoctoral and Early Career Researcher Exchange (PECE) bursary by the Scottish Informatics Computer Science Alliance (SICSA) for visiting Imperial College London in January–March 2016.





## 2 Content

The artifact package includes:

- the source code of the Scribble tool [1], extended with support for *type projection and merging* and Scala API generation, based on the theory developed in the companion paper (see, in particular, § 7);
- the source code of the `lchannels` library [2, 3];
- several examples of multiparty protocols (in Scribble notation), including the main running example of the companion paper;
- working Scala implementations of the aforementioned example protocols, written by using the Scribble-generated Scala APIs;
- a ready-to-use VirtualBox disk image containing an Ubuntu 16.04 installation, fully configured for testing our artifact. The disk image also includes an easy-to-use graphical tool for demoing the protocol examples above;
- an `index.html` file with detailed instructions describing the VirtualBox disk image, the graphical demo tool, the (extended) Scribble syntax, how to use the Scribble from the command line, how the various examples work, and how to navigate the implementation source code.

## 3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). The latest version of our code is available on following websites and repositories:

- Scribble:
  - main website: `http://scribble.org/`
  - repository with Scala API generation support: `https://github.com/alcestes/scribble-java/tree/linear-channels`
  - **NOTE**: the repository link above points to the `linear-channels` development branch. We expect that this branch will be eventually integrated in the main Scribble repository.
- `lchannels`:
  - website: `http://alcestes.github.io/lchannels/`
  - repository: `https://github.com/alcestes/lchannels`

## 4 Tested platforms

The artifact disk image is known to work on any platform running Oracle VirtualBox version 4 or 5 (`https://www.virtualbox.org/`) with 5 GB of free disk space and 2 GB of free RAM.

Scribble should compile on any platform running Java 8 and Maven 3.3 (`https://maven.apache.org/`), using the standard build procedure.

`lchannels` should compile on any platform running Java 8 and the Scala Build Tool 0.13 (`http://scala-sbt.org/`).

## 5 License

- Scribble is released under the Apache License version 2.0 (`http://www.apache.org/licenses/LICENSE-2.0.html`).
- `lchannels` is released under the BSD 2-clause License (`https://opensource.org/licenses/BSD-2-Clause`).

## 6 MD5 sum of the artifact

ae0ea460fbe40c7a96abba0913fe546c

## 7 Size of the artifact

1.6 GB.

**Acknowledgements** The authors wish to thank Sung-Shik Jongmans, Rumyana Neykova, Nicholas Ng, and Bernardo Toninho for testing the artifact, and the anonymous artifact reviewers for their comments and suggestions.

## References

1 Raymond Hu and Nobuko Yoshida. Hybrid session verification through endpoint API generation. In *FASE*, 2016. doi:10.1007/978-3-662-49665-7_24.

2 Alceste Scalas and Nobuko Yoshida. Lightweight session programming in scala. In *ECOOP*, 2016. doi:10.4230/LIPIcs.ECOOP.2016.21.

3 Alceste Scalas and Nobuko Yoshida. Lightweight Session Programming in Scala (Artifact). *Dagstuhl Artifacts Series*, 2(1), 2016. doi:http://dx.doi.org/10.4230/DARTS.2.1.11.