REIHE INFORMATIK
18/2001

**RTP/I Payload Type Definition for Application Launch Tools**

Jürgen Vogel
Universität Mannheim
Praktische Informatik IV
L15, 16
D-68131 Mannheim

# RTP/I Payload Type Definition for Application Launch Tools

Jürgen Vogel
Department for Praktische Informatik IV
University of Mannheim
vogel@informatik.uni-mannheim.de

**Abstract**

This document specifies an application-level protocol (i.e., payload type) for application launch tools using the Real-Time Protocol for Distributed Interactive Media (RTP/I). RTP/I defines a standardized framing for the transmission of application data and provides protocol mechanisms that are universally needed for the class of distributed interactive media. An application launch tool is used to synchronously start applications in collaborative environments, i.e., a participant can trigger the simultaneous execution of a program at all involved sites. This document specifies how to employ an application launch tool with RTP/I and defines application data units (ADUs) for application launch tool operations. This protocol definition allows standardized communication between different application launch tool implementations.

## 1 Introduction

*Distributed interactive media* are media which allow a set of spatially separated users to synchronously interact with the medium itself. Typical examples of distributed interactive media are shared whiteboards, which are used to present and edit slides in a teleconferencing environment [Gey99], as well as distributed virtual environments (DVEs) [Hag96], shared text editors [HC97], and computer games with network support [GD98]. Application launch tools also belong to the class of distributed interactive media. With the help of an application launch tool an application can be started simultaneously at all sites. This application can either be collaboration-aware or collaboration-unaware. For instance, in a teleteaching scenario the lecturer could launch a video clip or an animation in order to visualize a certain topic. Once started, however, the application can no longer be controlled by the application launch tool.

It should be noted that an application launch tool introduces a security risk into distributed environments, since remote participants can also start ”evil” programs. Thus, an application launch tool should take precautions to protect the local system.

In the following, we briefly introduce the class of distributed interactive media and the Real-Time Protocol for Distributed Interactive Media (RTP/I). For a more detailed discussion please refer to [Mau00, MHKE01, MHK+00].

In order to provide high responsiveness and to avoid the drawbacks of centralized approaches, such as the presence of a single point-of-failure and lack of scalability, applications for distributed interactive media often employ a replicated distribution architecture. In this architecture each user runs an instance of the application which manages a local copy of the medium’s shared state. For example, the state of an application launch tool comprises all information necessary to start an application such as program name, path, and parameters.

The state of a distributed medium changes either by the passage of time or by means of user interactions (*events*). State changes due to the passage of time can be calculated locally and need not be distributed among application instances. In contrast, events have to be exchanged among instances via the network to all remote instances of the application so that each can modify its local copy of the state accordingly. For better handling, the application’s state can be partitioned into several *sub-components*. Each application that can be started within an application launch tool session is represented by an RTP/I sub-component.

RTP/I is an application-level protocol that employs the media model described above, and is applicable to arbitrary distributed interactive media. It consists of two main parts; both reside on the application level and are independent of the underlying network and transport layers:

- the framing protocol (RTP/I). RTP/I is used to frame the data transmitted by distributed interactive media. The RTP/I framing contains the information that is common to media of a specific class. This information makes it possible to understand to a large extent the semantics of the transmitted data without any medium-specific knowledge. Therefore, meaningful functionality and services that are independent of the media-specific data encapsulated by the framing information can be developed.

- the RTP/I control protocol (RTCP/I). RTCP/I is used to convey meta information about the medium and information about the participants in a session.

RTP/I is not a complete protocol. It needs to be adapted to the requirements of a specific medium or a group of media by defining either a payload or a profile. A profile adapts RTP/I to the needs of a group of distributed interactive media. A payload type definition is a specification document that defines how a particular medium is transported using the framework provided by RTP/I. Essentially, it describes how the medium-specific data are encoded. The aim of such a standardized protocol is to allow communication between different application implementations. The specific encoding "application launch tool" as defined by this document is assigned the payload type "6". Each RTP/I ADU carries this payload type as the identifier of the originating application.

The remainder of this document is structured as follows. First, we explain how RTP/I can be used for application launch tools. Then we define all the necessary application data units (ADUs) of an application launch tool protocol. These ADUs are transported either as an RTP/I state or event.

# 2 Usage of RTP/I

The state of an application launch tool consists of information about all applications that can be started at a certain point in time. This information includes a describing name (e.g., "flow control animation"), the program name (e.g., "animation.exe"), and program parameters (e.g., "-p 8000"). In contrast, the path where the application is stored is not part of the shared state, since this path might differ at the participating sites. Thus, the path is assigned only locally, and the application launch tool should make sure that it is able to find the specified application via the local path. In the following, the set of describing name, program name, and parameters is called an application.

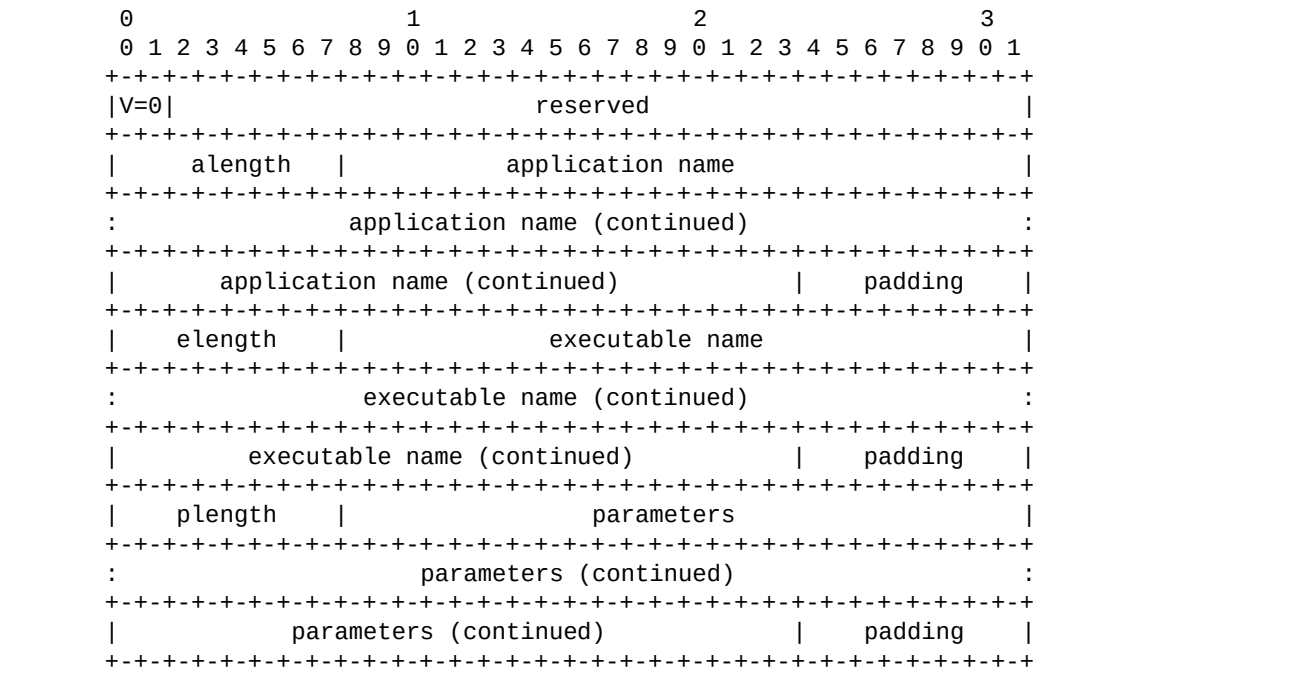
Each application is represented by an RTP/I sub-component. Each sub-component is assigned a unique ID that is allocated by a unique number service [MHK+00]. The creation of a new application is signaled by a state transmission, whereas the payload of the RTP/I state encodes the application information. State changes (e.g., altering the parameters or starting an application) are encoded as RTP/I events.

Each participant is assigned a unique identifier (participant identifier, pid). The pid can be used to identify the participant that has started an application. Management of identifiers is beyond the scope of this document, please refer to [MHK+00].

This document does not specify a certain transport protocol. Rather, it is assumed that the application makes use of a reliable transport mechanism that guarantees the reliable distribution of operations. This mechanism can be integrated either into the application or into RTP/I, or can be implemented at the transport level.

# 3 Application Launch Tool ADUs

## 3.1 Application State ADU




version (V) : 2 bits
     This field specifies the version number of this protocol. The version
     defined by this specification is version 0.

reserved : 30 bits
     These bits are reserved for future use.

application name length (alength) : 8 bits
     This field identifies the length of the application name in bytes
     excluding any padding bytes.

application name : alength bytes + padding
     This field contains the application name as ascii text. If the application
     name length is not a multiple of 4 bytes, a minimum number of the padding
     octets necessary to pad the application name to a full 32-bit boundary
     follow the application name.

executable name length (elength) : 8 bits
     This field identifies the length of the executable name in bytes
     excluding any padding bytes.

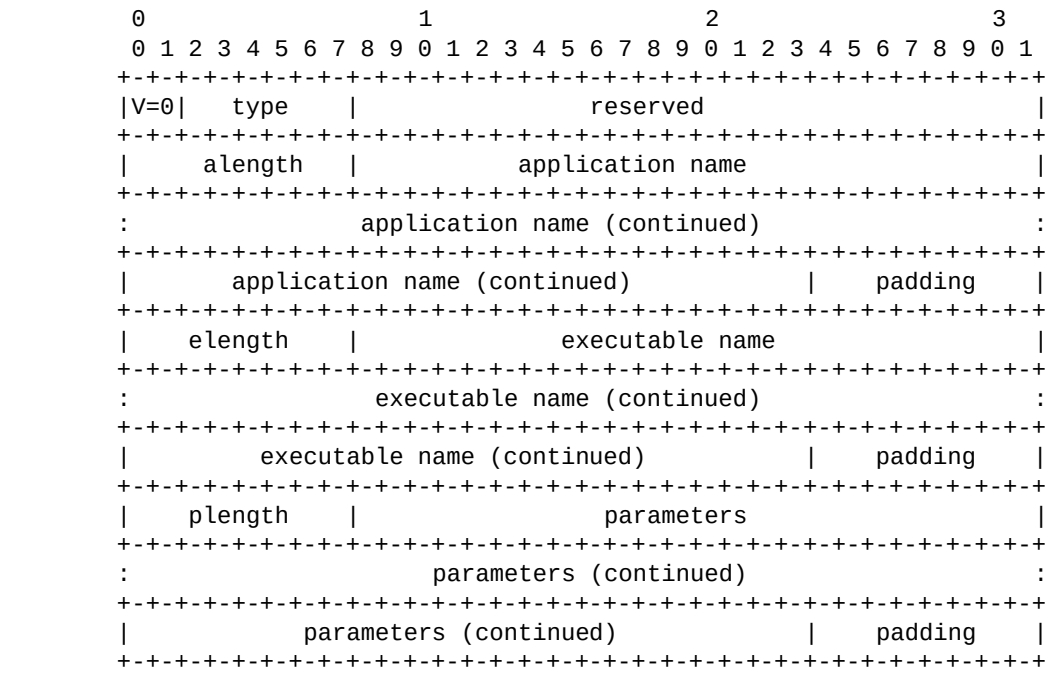
executable name : elength bytes + padding
     This field contains the executable name as ascii text. If the executable
     name length is not a multiple of 4 bytes, a minimum number of the padding
     octets necessary to pad the executable name to a full 32-bit boundary
     follow the executable name.

parameters length (plength) : 8 bits
     This field identifies the length of the parameters in bytes
     excluding any padding bytes.

parameters : plength bytes + padding
     This field contains the program parameters as ascii text. If the
     parameters length is not a multiple of 4 bytes, a minimum number of the padding
     octets necessary to pad the parameters to a full 32-bit boundary
     follow the parameters.

## 3.2 Event ADUs

### 3.2.1 Edit Application ADU




version (V) : 2 bits
     This field specifies the version number of this protocol. The version
     defined by this specification is version 0.

type : 6 bits
     This field identifies the type of event. It is set to 0 (edit application).

reserved (res) : 24 bits
     These bits are reserved for future use.

application name length (alength) : 8 bits
     This field identifies the length of the application name in bytes
     excluding any padding bytes.

application name : alength bytes + padding
     This field contains the application name as ascii text. If the application
     name length is not a multiple of 4 bytes, a minimum number of the padding
     octets necessary to pad the application name to a full 32-bit boundary
     follow the application name.

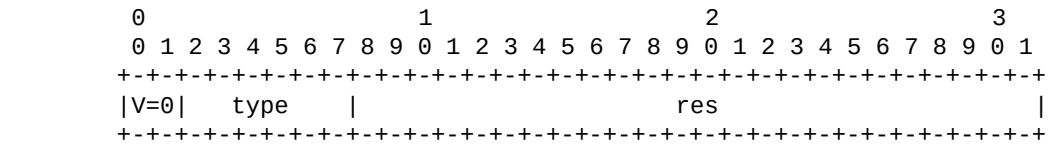
executable name length (elength) : 8 bits
     This field identifies the length of the executable name in bytes
     excluding any padding bytes.

executable name : elength bytes + padding
     This field contains the executable name as ascii text. If the executable
     name length is not a multiple of 4 bytes, a minimum number of the padding
     octets necessary to pad the executable name to a full 32-bit boundary
     follow the executable name.

parameters length (plength) : 8 bits
     This field identifies the length of the parameters in bytes
     excluding any padding bytes.

parameters : plength bytes + padding
     This field contains the program parameters as ascii text. If the parameters
     length is not a multiple of 4 bytes, a minimum number of the padding octets
     necessary to pad the parameters to a full 32-bit boundary follow the parameters.

### 3.2.2 Delete Application ADU

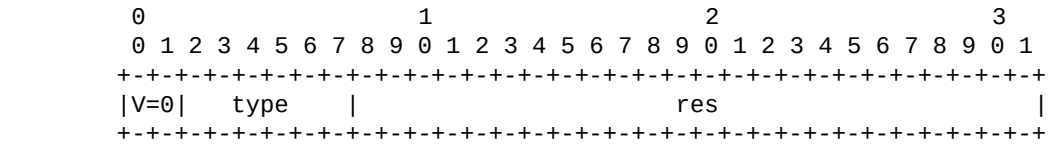



version (V) : 2 bits
: This field specifies the version number of this protocol. The version defined by this specification is version 0.

type : 6 bits
: This field identifies the type of event. It is set to 1 (delete application).

reserved (res) : 24 bits
: These bits are reserved for future use.

### 3.2.3 Launch Application ADU




version (V) : 2 bits
: This field specifies the version number of this protocol. The version defined by this specification is version 0.

type : 6 bits
: This field identifies the type of event. It is set to 2 (launch application).

reserved (res) : 24 bits
: These bits are reserved for future use.

# References


[GD98] L. Gautier and C. Diot. Design and Evaluation of MiMaze, a Multi-player Game on the Internet. In *IEEE International Conference on Multimedia Computing and Systems*, pages 233–236, 1998.

[Gey99] W. Geyer. *Das digital lecture board – Konzeption, Design und Entwicklung eines Whiteboards für synchrones Teleteaching.* PhD thesis, University of Mannheim, Germany, 1999.

[Hag96] O. Hagesand. Interactive multiuser VEs in the DIVE system. *IEEE Multimedia*, 3(1):30–39, 1996.

[HC97] M. Handley and J. Crowcroft. Network Text Editor (NTE) – A scalable shared text editor for the MBone. In *ACM SIGCOMM*, pages 197–208, 1997.

[Mau00] M. Mauve. *Distributed Interactive Media*. PhD thesis, University of Mannheim, Germany, 2000.

[MHK+00] M. Mauve, V. Hilt, C. Kuhmünch, J. Vogel, W. Geyer, and W. Effelsberg. RTP/I: An Application-Level Real-Time Protocol for Distributed Interactive Media. Internet Draft: draft-mauve-rtpi-00.txt, 2000.

[MHKE01] M. Mauve, V. Hilt, C. Kuhmn̈ch, and W. Effelsberg. RTP/I - Toward a Common Application-Level Protocol for Distributed Interactive Media. *IEEE Transactions on Multimedia*, 3(1), 2001.