A Platform-Independent Token System for Payment Terminals

Elizabeth D. Rather
erather@forth.com
Manhattan Beach, CA

Stephen Pelc
sfp@mpeltd.demon.co.uk
Southampton, England

Peter Johannes
pjo@europay.com
Waterloo, Belgium

Europay, the major European credit card organization (Eurocard, MasterCard in Europe, and other financial products) is developing technology to support smart cards—Integrated Circuit Cards, or ICCs—as the credit cards of the future. This will require new software in all credit card terminals, which range from 8051-based POS terminals to high-end ATMs. To facilitate this transition, they have designed a token-based system—conceptually similar to Open Firmware or Java—which is based on Forth. Using this Open Terminal Architecture (OTA), it will be possible for credit card issuers and acquirers to write application programs that will be completely platform independent and which will run on all OTA-compliant kernels.

The project has been under way for two years. FORTH, Inc. and MPE, Ltd. have been principal members of the design and development team, along with Europay. Prototype terminals were exhibited at a major Europay banking conference in June 1996, and production systems have been operating in the field in Prague, Czech Republic, since May 1997.

Background

Modern payment applications are moving to ICC technology. ICCs can significantly improve security of payment transactions by being able to manage encrypted account data offline, by participating actively in the transaction-validation process, and by being intrinsically extremely difficult to violate or reproduce. They can also contain code to enhance the transaction processing, thereby providing new opportunities for payment products and services.

Use of this new technology, however, will necessitate altering the firmware in several million terminals that will use the ICCs. To facilitate this transition, Europay is designing a standardized software system that will be compact, efficient, and easy to maintain and enhance for future payment system needs. This is the Open Terminal Architecture system.

OTA defines a software virtual machine standardized across all terminal types, described in detail in *Europay Open Terminal Architecture Specification Volume 1: Virtual Machine Specification*.[1] This virtual machine provides drivers for the terminal's I/O and all low-level CPU-specific logical and arithmetic functions. An extensive repertoire of commands specific to the needs of ICC terminals is also provided, with functions such as commands for managing databases, different languages, security algorithms, and the special data formats used by the cards. High-level libraries, terminal programs, and payment applications using standard kernel functions may be developed and compiled into token modules. These must be certified once; thereafter, they will run on any conforming terminal of the appropriate type (for example, ATM or POS) without change, regardless of the terminal's CPU type or other architectural issues. Therefore, a significant consequence of OTA is a simplified and uniform set of test and certification procedures for all terminal functions.

To provide a common means of distributing programs in a compact, standard, machine-readable form, OTA uses a token system that is, in some respects, similar to Java byte-codes. An OTA token compiler converts source code to a string of tokens that is extremely compact (and therefore easy to transmit over phone lines or to read from an ICC), and is also easy for even simple processors to interpret with minimal overhead.

To summarize, OTA provides the following major benefits:

- A virtual machine with generalized ICC support functions, to be installed in each terminal only once. The kernel lifetime is expected to match that of the terminal (7–10 years).
- Terminal kernel certification independent of applications, so certification only needs to be done once for each terminal type. A terminal type is defined as a specific configuration of terminal CPU and I/O functions.
- Application certification procedures that are independent of the terminal on which the application will run, since all terminals provide the same virtual machine interface. Only one certification and validation is needed for tokenized software libraries, terminal programs, and payment applications, providing they run on certified OTA terminals.
- Standard downloading procedure for all terminal types, using compact token modules for minimum transmission time.
- Support for tokenized code on an ICC, to make maximum use of its storage capabilities and to minimize communications time between card and terminal.

OTA is based on Forth, extended with commands to facilitate development of payment applications. Forth was chosen by Europay because, of all standard interpretive-type languages, it provides the most compact and efficient means of representing both terminal programs and the code that may reside on the ICC itself. Compactness in terminal programs translates directly into reduced transmission time and cost for terminal updates, and compactness in ICC code results in increased capability and reduced transfer time between card and terminal.

For security reasons, OTA allows only run-time behavior in a terminal, so the virtual machine includes only a run-time subset of ANS Forth.

Both Forth and C compilers have been developed to support OTA tokens. VM implementations, applications, and libraries have been developed in both Forth and C.

Open Terminal Architecture Features

The specific characteristics of the architecture were designed and optimized for both compact and reasonably fast execution of typical payment functions on a wide variety of

1. Version 2.2, January 29, 1997. Available from Europay Documentation Centre, 198A Chaussée de Tervuren, 1410 Waterloo, Belgium.

CPUs. Many design decisions were heavily influenced by the extreme need for program security in payment terminals.

Virtual Machine CPU

The OTA virtual machine is based on a multi-stack architecture, as seen in Figure One. This architecture, derived from Forth, has been further modified for portability, code density, ease of compilation, and for use with other programming languages. For example, it contains frame memory for local variables used in C. Thus, OTA token compilers can be written not only for Forth, but also for C and other languages.

The VM is a byte-addressed, 32-bit machine, with 32-bit registers and stack elements. Despite some initial trepidation about implementing a 32-bit VM on processors such as the 8051, we have found ways to do so with remarkably good run-time performance.

Memory

OTA defines a single address space for programs. This address space is accessible for data storage only. Programs may not assume that executable code is in this address space. Depending on the actual processor, and on the mechanism used to convert the token image into executable tokens, the executable code may be in a different address space (shown as code space in Figure One), or may be under the control of a memory management unit. In any case, programs are not permitted to access their own program memory directly, and any attempt to do so will be flagged during the program certification procedure.

Addressable memory is further divided into sections:

- Initialized data space may be preset at compile time to values that will be instantiated in the target at run time.
- Uninitialized data space will be preset to binary zeroes in the target at run time.
- Extensible memory is temporarily allocated using a rubber-band memory allocation algorithm.
- Frame memory is used by C stack frames and Forth local variables.

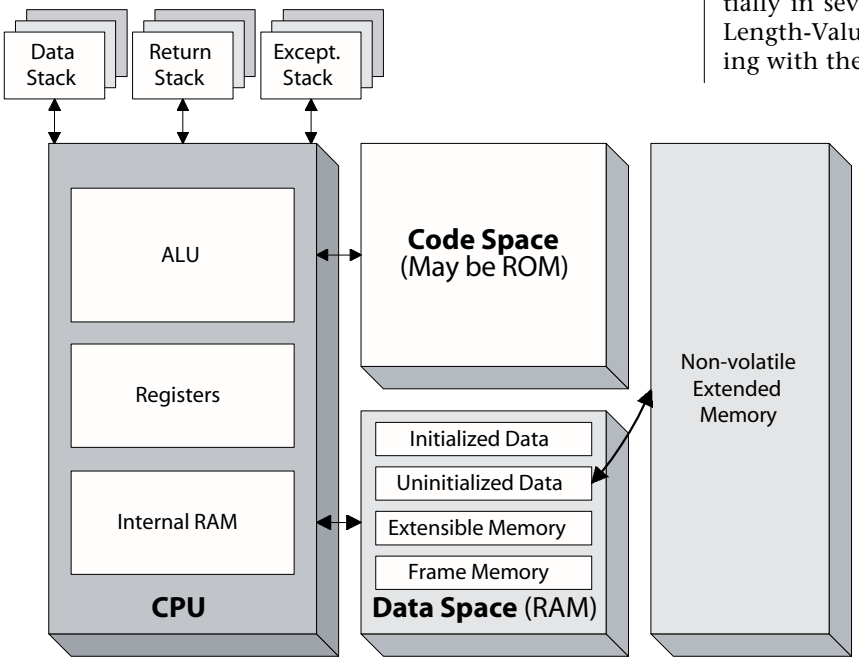
In addition to directly addressable memory, the VM also manages extended memory, which is not directly available to token programs. This is used for two purposes: databases and module storage. Databases are managed by the VM as a server to client token programs. Clients may select a named database and records within that database. At any time, the client has access to a current record in a current database via named fields of various types. The module storage is not accessible by token programs directly, although modules may call functions in external modules in ways managed internally by the token interpreter in the VM. Certain tokens also support high-level management of the module storage by allowing terminal programs to add and delete modules, with appropriate security.

Programs and Tokens

The OTA token set provides program portability across multiple CPU types by passing source code programs of various types through a compiler whose output is a string of OTA tokens, which may be thought of as machine instructions for the OTA virtual machine. The tokens are organized into a module, which consists of a header, a section representing the module's data items, lists of imported and exported functions (providing links to other modules), and the tokens themselves. Target terminals then process this code by instantiating the data space associated with the module, linking the module's imports to functions exported by other modules, and finally interpreting the tokens. Figure Two illustrates this process.

The OTA token set covers three main areas. The first is the instruction set of a theoretical processor (the virtual machine), which provides the instructions necessary for the efficient execution of programs. The second supports I/O and communications functions. The third group consists of OTA-specific functions such as databases, a message database (potentially in several European languages), and support for Tag-Length-Value data formats (ISO 8825) used for communicating with the ICCs and for other data communications.

Figure One. The OTA Virtual Machine



The OTA token set has been optimized for use on small terminals, with ease of compilation, ease of interpretation, and good code density. The most common functions, including most Forth primitives, are expressed in one-byte, or primary, tokens. Less frequently used functions are two-byte, or secondary, tokens. Some tokens also have associated values, for such things as literal values and branch offsets.

System Components

The purpose of OTA is to provide software to run in terminals used in payment applications. Conceptually, there are two hardware environments, and several classes of software. The hardware environments include the development system, which is based on a simple PC; and a target, which is some form of payment terminal. The entire

suite of software includes:

- development software, which runs on the PC and is available in two packages, for VM and application development, respectively;
- virtual machine implementations, which include all platform-specific software in a terminal and other mandatory standard functions;
- libraries, which provide general functions to support terminal programs and payment applications;
- applications, which are the functions specific to a particular payment product;
- terminal programs, which perform general non-payment terminal functions and include high-level mechanisms for selecting and executing transactions and associated applications; and
- test suites and platforms, for both VM implementations and token programs.

Terminal Target Environments

The target system is any one of a large variety of payment terminals. Actual products range from small, hand-held devices with simple, eight-bit microprocessors (such as the 8031/51 family), to 32-bit computers running operating systems such as Windows NT. In order to simplify the production, certification, and maintenance of software on such a wide variety of targets, OTA terminal code is based on a single virtual machine. The VM consists of a standardized set of functions whose CPU-specific implementation is optimized for that specific platform. Implementations currently operating in the field on eight different devices show that this approach provides good run-time performance, even on 8051 CPUs.

Virtual Machine

The OTA VM has standard characteristics that define addressing modes, stack usage, register usage, address space, etc. The virtual machine concept makes a high degree of standardization possible across widely varying CPU types, and simplifies program portability, testing, and certification issues.

The VM instruction set includes a selected subset of ANS Forth commands, plus a number of specialized OTA functions, such as terminal I/O support and token loader/interpreter support. Since it cannot itself be tokenized, and may reside in PROM, the VM is intended to be installed once, and not changed thereafter during the lifetime of the terminal. Therefore, its functions are carefully designed to be very general in nature and as complete as possible, in order to support a wide range of present and future terminal programs and applications.

Terminal manufacturers are responsible for providing a VM implementation on their terminals. This VM is developed and certified according to the OTA Virtual Machine Specification. Standard kernel functions not appropriate to a particular terminal type (e.g., the cash dispenser function on a POS terminal) are coded as null functions for that terminal, so every kernel has an identical set of functions and the testing and certification process is simplified. These null functions add very little to system overhead and complexity, and their advantage far outweighs their cost.

The terminal's VM supports standard libraries and terminal programs and applications, which are written in high-level code for the virtual machine and are delivered as token modules, which will run on any standard VM.

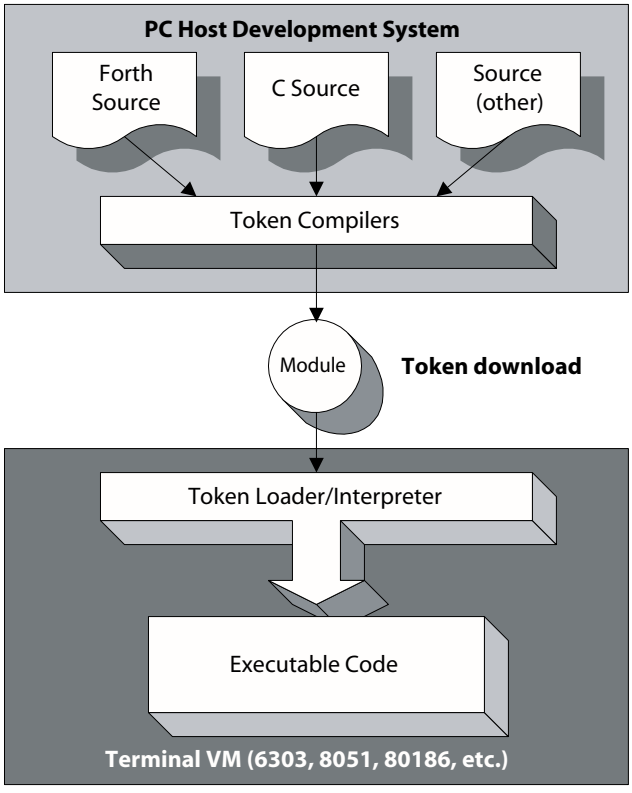
Libraries

OTA libraries contain higher-level functions that support common features of terminal programs, such as language selection, and common features of applications, such as PIN verification. A terminal may contain several libraries, some accessible to all applications, and some restricted to particular applications or payment systems. Libraries are written and tokenized for the virtual machine, using functions provided in the kernel, and therefore can be run on any terminal.

Terminal Program

A terminal program consists of the high-level personality characteristic of this terminal type (POS, ATM, etc.). This includes the functions common to all transactions (e.g., card initialization and language selection), as well as the user interface required to select an application and process a transaction. The terminal program, at the highest level, is typically triggered by a card insertion. A terminal program is written for the virtual machine and is supplied in token form. It can, therefore, be run on any terminal of the appropriate type, and is easily changed by downloading over a network at any time. However, it frequently does incorporate platform-specific features, such as customized greeting messages, knowledge of particular screen-management capabilities, etc. These features are not included in the VM, as they may change on a time-scale shorter than the design lifetime of the VM. For example, a particular make of terminal may be used by a number of different merchants, each of which may request customized user menus.

Figure Two. Tokens may be generated from a variety of source formats, downloaded to a terminal, and then converted into executable code for that terminal by any of several different methods.



Applications

A terminal transaction will select an application as part of its processing flow. Applications fall into three general areas: stored value system (such as Europay's CLIP system, VISA CASH, or Mondex), debit cards, and credit cards; applications generally will vary in their method of processing a given transaction. Versions of these applications may be provided by different payment systems and may be further customized by individual issuers, acquirers, or even individual merchants (such as large chains or department stores). Applications are supplied in token form via the communications path and, if security considerations permit, may be enhanced by token programs on an ICC.

The Token Compiler and Token Loader/Interpreter

Libraries, applications, and terminal programs are written in high-level code for the virtual machine. The OTA development system includes a special compiler for this virtual machine, whose output consists of tokens. Tokens may be thought of as machine language instructions for the virtual machine. Tokens are either one or two bytes in length, and therefore represent the program in a form that is both CPU-independent and extremely compact (far more so, for example, than compressed source text).

Each OTA virtual machine contains a token interpreter (TLI), which processes a stream of tokens into an executable form. Once the kernel is installed in a terminal, the libraries, applications, and terminal programs can be downloaded into the terminal in a variety of ways (direct connection to an OTA development host, acquirer network, modem and dial-up telephone line, ICC, etc.). Program modifications and entire new applications may be downloaded in the same manner whenever needed. The VM implementation is designed to be so general purpose in nature that a wide range of present and future terminal programs and applications can be accommodated without modifications to the VM.

ICC Functions

One function of ICCs is to improve transaction security by incorporating and managing encrypted data and participating actively in the transaction-validation process. It is a natural feature of OTA to go beyond these functions and to provide for ICCs that also contain program code to enhance a terminal's transaction processing, thereby providing new opportunities for payment products and services. To facilitate this, a few sockets have been provided that can be plugged by issuer-specific functions such as loyalty programs, which may be invoked at appropriate points in the transaction processing. Europay does not currently propose that ICCs contain entire applications, but only plug behaviors that enhance existing terminal applications.

As far as security is concerned, the presumption is that if an ICC passes the decryption and data authentication tests performed by the terminal program, whatever functions and plug behaviors are on the card have been certified and are syntactically valid. The terminal decides to allow or disallow the card's proposed actions only as controlled by the terminal access security functions.

Development Environments

An OTA development system is used to develop terminal software, either low-level VM implementations or high-level library or application software. Kernel development requires a target terminal to be connected, as the kernel is cross-compiled on the PC host and downloaded to the terminal across the Interactive Development Link. OTA libraries, terminal programs, and applications are also developed on the PC host. Because they are high-level code, they may also be executed on the host for preliminary testing, using a PC version of the standard kernel.

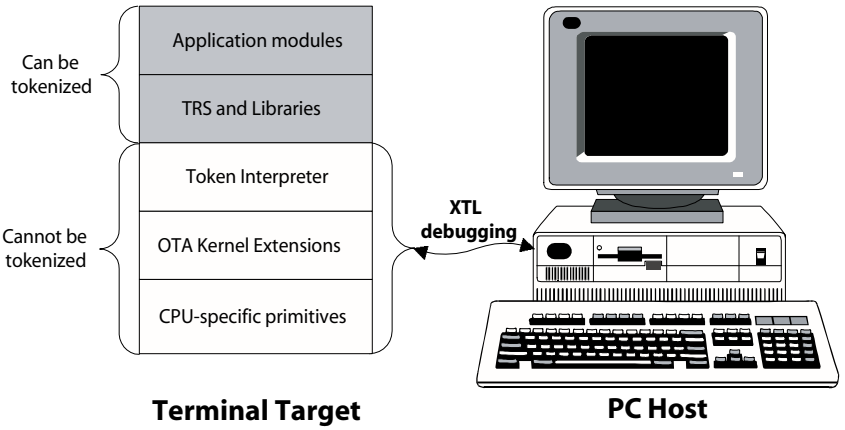
Since the requirements for developing and testing the VM implementation and high-level token modules differ significantly, two different tool chains have been developed: the Kernel Development Kit (KDK) for terminal VM implementors, and the Application Development Kit (ADK) for token program developers.

VM Implementation Tools

The KDK consists of a Windows-based cross-compiler and Interactive Development Environment (IDE), a test terminal consisting of a CPU, keypad, LCD display, serial ports, and memory sufficient to run a typical ICC application supplied in token form as a demo, and a VM implementation for that test terminal hardware (see Figure Three). Development software on the PC is based on ProForth for Windows, a product of MicroProcessor Engineering Ltd.

The VM implementation for the test terminal is supplied in source form, with documentation intended to support a developer porting this to a specific terminal of the same CPU type. As many terminal vendors have previously developed BIOS or OS functions (typically in C or assembler), a protocol is included that facilitates the use of this software to provide a defined set of functions (e.g., I/O functions) required by the VM. The kit also includes a terminal test suite and demo

Figure Three. A typical OTA development environment for a terminal. A small program to support the terminal end of the Cross-Target Link (XTL) protocol is included in the target during development.



application for testing the VM as it develops.

Token Module Development Tools

There are two major components to the Application Development Kit. These are the token compiler itself, and the Token Interactive Debugging Environment (TIDE). The latter is a standard VM implementation on a Windows platform, with configuration options that enable it to simulate a wide variety of potential target terminals. A developer may use this platform to test token modules, regardless of the language in which they were written. Token compilers are available for Forth and C. The ADK also includes an optimizer, which performs a variety of post-processing functions on token modules, that can reduce a module's size by on the order of 50%. The result is a module that runs with no performance penalty; indeed, it is usually significantly faster, since there are fewer tokens to process.

Testing and Validation Tools

VM test suite development has progressed in parallel with VM development, and test suites are presently delivered with all KDKs. The VM test suite consists of a set of modules that exercise individual tokens, checking each token's behavior against expected results and producing a report summarizing tests passed and failed (if any). The main test platform for token modules is TIDE, although tools are also in development to provide additional test facilities.

Formal validation tools are still in development.

Project Status

The Open Terminal Architecture (OTA) is a complete system for supporting ICC-based payment terminals of the future. Its design incorporates features that will facilitate development and certification of a new, standardized kernel for all payment terminals; will support development and certification of platform-independent libraries and terminal programs; and will enable code on the ICCs themselves to provide enhanced payment products for issuers.

Extensive effort has been directed toward seeing that both code and procedures are included to ensure program security and integrity. A complete set of development tools is available through Europay to support not only Forth, the language upon which OTA is based, but also C. Reference kernels, libraries, and applications are also available.

Elizabeth D. Rather was the world's second Forth programmer. She is President of FORTH, Inc., and a member of the Board of Directors of the Forth Interest Group.

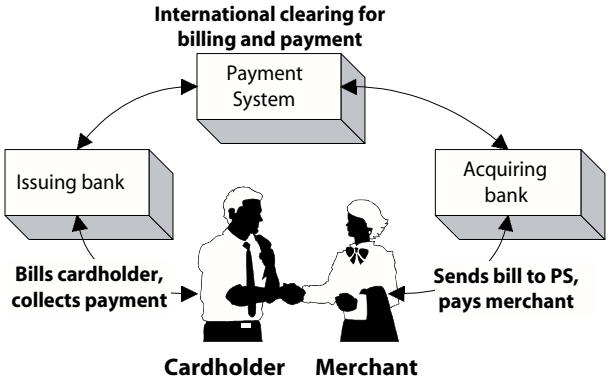
Stephen Pelc's company, Microprocessor Engineering Ltd., has been selling Forth-based hardware and software since the early 1980s.

Peter Johannes has been the project manager for the Open Terminal Architecture since its inception in 1994.

Background:

Payment Systems in Europe

Banking-based payment systems in Europe are organized nationally for domestic banking products, and internationally to facilitate international consumer payments. Looking at a typical payment transaction, the parties involved are the merchant, the acquiring bank, the payment system, and the issuer. Each of these is a distinct role, although it need not be played by a different physical or corporate entity.



- Issuers provide consumers with banking products, in this context as debit, credit, or stored-value cards. These cards can still be embossed only (for paper-based transactions), with magnetic stripe (today's state-of-the-art) or with an IC. IC cards are currently used widely only in France, although pilot projects are under way in several other countries.
- Acquirers hold the commercial relationship with the merchant. They acquire the transactions, meaning they pay the merchant for the goods and route the transactions to wherever the consumer's account is.
- Merchants accept cards, and provide goods to the cardholder. Merchants are more and more convinced of the necessity to migrate toward electronic payment services (e.g., magstripe or ICC).
- Payment systems such as Europay have two primary roles: to provide specifications (and associated certification) for international payment products, and to arbitrate member conflicts. In addition, payment systems also typically provide network and security services.

An IC card (or ICC) is an active device that stores data in such a way that it can prove their authenticity. It is also capable of generating *authenticity certificates* for transactions. This means that a point-of-sale terminal now has to deal with an active device that performs complex operations. The terminal software must behave correctly, whichever ICC is used.

Current ICCs carry about 200 times the amount of information that can be present on the magstripe cards. This amount is expected to increase as ICCs get loaded with more services. The drawback is that terminal programs become more complex and bigger. Terminals typically download their software from a server via a 1200-baud communication link. The cost of this management and download is becoming prohibitive, especially in Europe.

Many existing ICC-based payment systems are "closed" (proprietary) systems. This means it is very hard to introduce new applications without getting the consent of all the parties involved. Even then, extensive changes to the software on every terminal are required, and the terminals must be re-certified. With OTA, Europay is attempting to provide an open system, capable of handling multiple ICC products, card types, and systems on a single terminal with minimal changes.

—P. Johannes, Europay International