

# Electronic Component Information eXchange

## QuickData Protocol Specification

Version 2.0-050100

Copyright © 2000 Silicon Integration Initiative, Inc. All rights reserved worldwide.

### Introduction

This specification defines an application level protocol for business-to-business (B2B) e-commerce exchange of electronic component (product) information. The protocol is specified by conceptual models, interchange formats, implementation constraints, and other semantics.

### Requirements

This specification solves specific business and technical problems and supports particular use models and other requirements, which are documented in “ECIX II Requirements”.

### Compliance

An implementation that fails to satisfy any constraint identified herein, or in the relevant normative references, with the keywords “required”, “shall”, or “shall not”, is not compliant to the specification. These keywords, along with the keywords “may”, “should”, “should not”, and “optional”, are to be interpreted as described in RFC 2119.

### Notation Conventions

This specification uses the following conventions:

1. Characters in `courier` are literal values to be used in code or messages exactly as shown.
2. Characters in `times` are not to be used literally but instead describe how values shall be constructed when used in code or messages.
3. Information contained in the following notation shall be considered non-normative:

(NOTE: This text is informative only and is not considered part of this specification.)

(EXAMPLE: This text is informative only and is not considered part of this specification.)

4. Characters bounded by double quotes have a special significance referenced as a whole in the sentence, such as a title, value, name, or other identifier.
5. An identifier bounded by angular brackets refers to a DTD element object. For example, <value> refers to the particular DTD element named “value”. If the letter “s” immediately follows the trailing right angle bracket it indicates one or more such elements.
6. An identifier bounded on the left by the percent character and on the right by the semicolon character refers to a DTD entity object. For example, %xml.att; refers to the DTD entity named “xml.att”.
7. An identifier bounded by single quotes refers to a DTD attribute object. For example, ‘NoValueReason’ refers to the DTD attribute named “NoValueReason”.

## References

### Normative

Documents identified as "Normative References" are referenced in part or in their entirety in the current document and the parts referenced shall have normative impact on compliant implementations. For each reference the edition indicated is the authority. All normative documents are subject to revision at different times and by different agencies. Consequently, application of more recent editions of such documents may affect interoperability of implementations and may have additional implications.

### Informative

Documents identified as "Informative References" are relevant to the current document, and may assist in comprehension of it, but do not have normative impact on compliant implementations.

## Terminology

protocol	Rules, guidelines, procedures, or formats by which data is exchanged between between two communicating parties.
semantics	Specific meanings or significance.

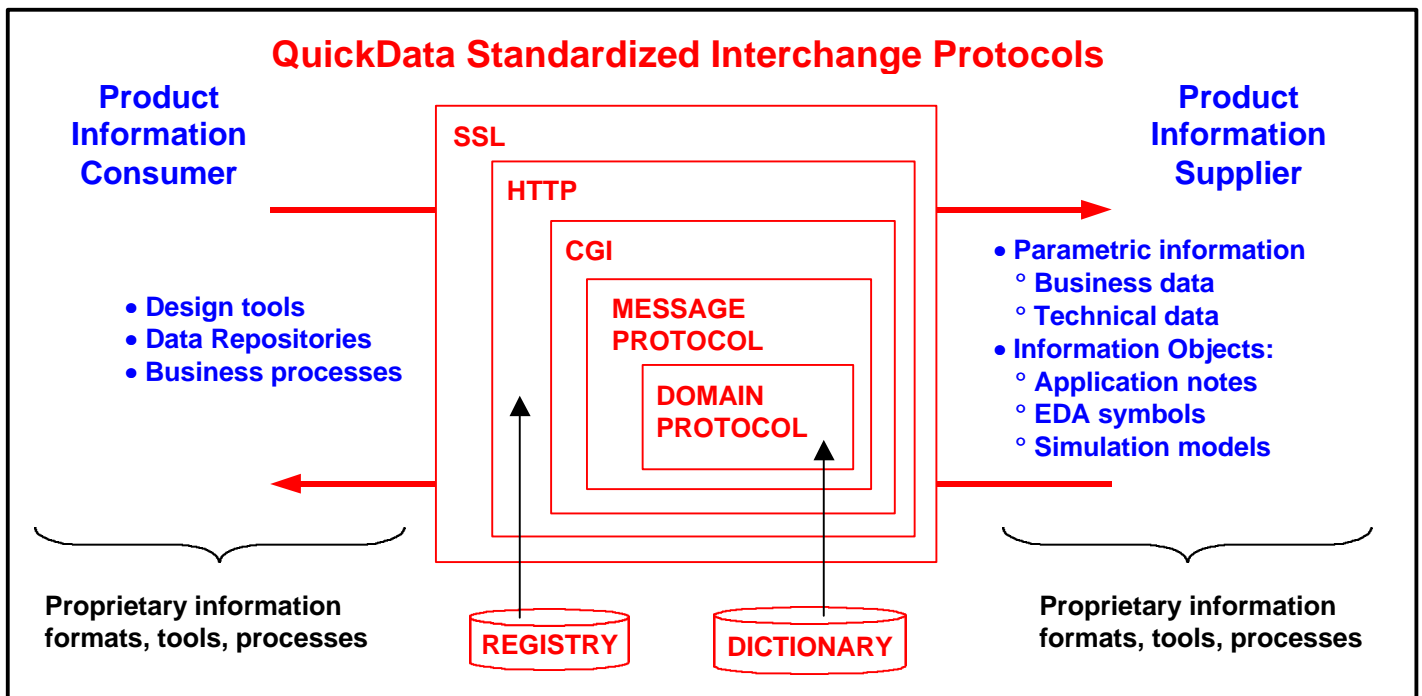
## Architecture

The architecture of the QuickData Protocol is built on existing protocols and supports modular exchange of product information using a real-time, query/response paradigm. Product information includes:

1. **Property data.** These include operational, functional, application, or other kinds of product characteristics. Property data is identified to the level of granularity of individual values that have sufficient machine-sensible semantics to support automated processing of product information data.

- 2. Information objects.** Product information objects of arbitrary content and structure can be transferred. Enough semantics are identified to allow unambiguous identification of the nature of the object, its format and encoding, and the types of tools that can process it.

**Figure 1: QuickData Architecture**



As Figure 1 illustrates, QuickData Protocol standardizes only the mechanisms for information interchange. Suppliers and consumers retain their own, proprietary, information schemas and processes.

A high degree of automated processing is achieved via a:

- standard XML message structure that allows identification of information semantics to a detailed level of granularity.
- Standard, computer sensible dictionary that mandates machine-sensible semantics and constraints for the information contained in a message.
- global registry of participants that allows automated discovery of current targets for interchange and dynamic establishment of connections to them.

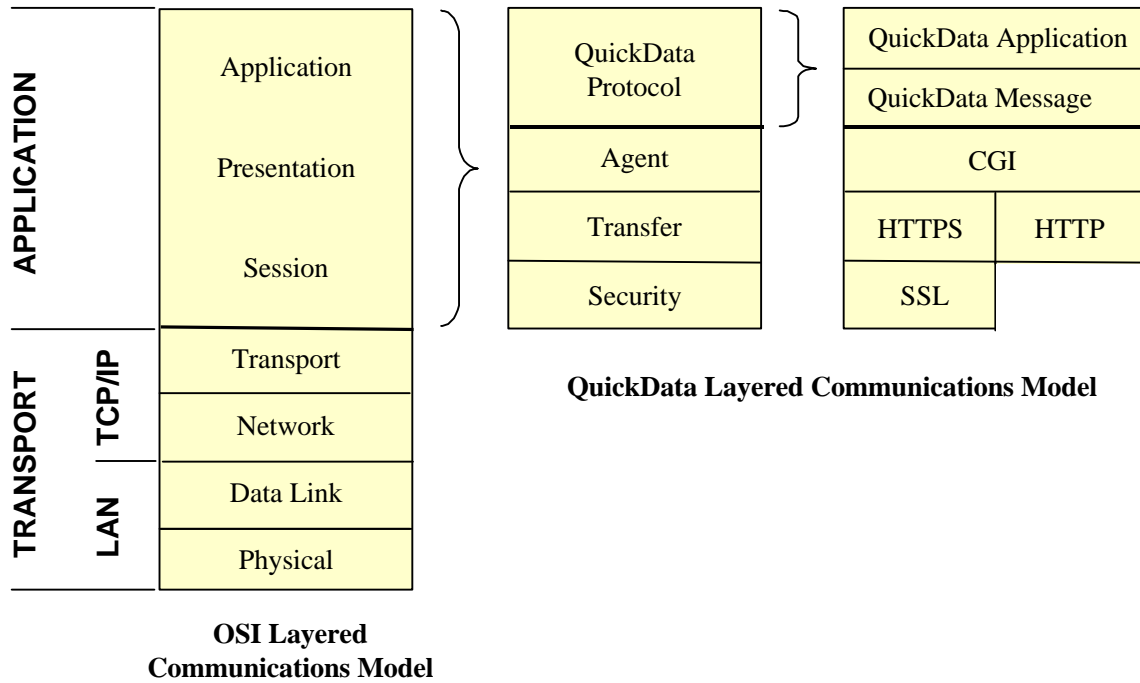
The QuickData Protocol Specification imposes constraints on all of the application layers of the OSI layered communications model, and defines the specifics of the QuickData Protocol layer.

## Protocol Layer Requirements

Communications protocols typically are layered into sets of logically separable functional components. This

layering allows the implementation of any given layer to be changed without affecting the layers above or below because the functional definitions and interfaces of the each layer are clearly defined. Figure 2 illustrates the QuickData definition of the Application layers of the OSI Layered Communications Model.

**Figure 2: Layered Communications Protocols**



This specification addresses dependencies, requirements, and definitions within the context of a layered protocol, based on the OSI model. QuickData divides the OSI Application layer into 5 sub-layers:

- Security,
- Transfer,
- Agent,
- QuickData Message, and
- QuickData Application Domain.

QuickData defines specific constraints for each of the “Application” layers.

## Security

Secure Internet communication, when used, shall be SSL V 3.

## Transfer

HTTP 1.0 or higher shall be used for transfer of messages.

## Agent

The Common Gateway Interface (CGI) protocol shall be used for encapsulating QuickData messages. In all cases the standard HTTP status codes shall be used.

### 1. If the message is a query:

**Header:** In addition to other header information required by CGI or HTTP, the following fields shall be set:

```
Content-type=application/x-www-form-urlencoded
Method=POST
```

**Entity body:** The entity body shall consist of:

```
QD=the_urlencoded_ECIX_II_message
```

In other words, the “NAME” attribute shall be “QD” and the “VALUE” attribute shall be a url-encoded XML instance that conforms to a Document Type Definition (DTD) with a valid QuickData formal PUBLIC identifier. This XML instance shall contain a <query> element.

### 2. If the message is a response to a query:

(a) In the case where the query does not contain a <data.object> element:

**Header:** Content-type=application/x-QD

**Entity body:** There shall be an XML instance that conforms to a DTD with a valid QuickData formal PUBLIC identifier. This instance shall contain a <response> element.

(b) In the case where the query does contain a <data.object> element:

i. In the case where a supplier returns the object:

**Header:** Content-type= the MIME type for the object being returned.

**Entity body:** The actual object only. The object is not imbedded in QuickData XML.

ii. In the case where a supplier does not return the object:

The requirements are the same as those for the case where the query does not include a <data.object> (2a).

## QuickData Message

The QuickData Message layer consists of constraints and semantics common to all application domains that use it. This layer is specified by the “QuickData Message Specification”.

## QuickData Application

The QuickData Application layer consists of constraints and semantics specific to a particular domain of use. The same QuickData Message protocol can be applied to different domains of use, that is, it can be used to exchange information within different business and technical contexts. (EXAMPLE: One QuickData Application is the “Electronic Components Domain Specification”.)

## ECIX Product Information Objects

The QuickData protocol can also be used to exchange product data of arbitrary internal structure, file format, and encoding. Such product information objects are defined by their own specifications, and if XML may have one or more DTDs and constraint specifications. (EXAMPLE: EDA Symbol, EDA Footprint.)

## Registry

The Registry contains machine-sensible information related to the participants in an exchange of information, thereby supporting fully automated negotiation of communications connections. The “Si2 Registry Specification” defines constraints and semantics common to all uses of the Registry.

## Dictionary

The dictionary contains human and machine-sensible semantics and constraints on the information in a message, thereby standardizing the interpretation of that information. The “Electronic Components Technical Dictionary Specification” defines constraints and semantics common to all uses of the dictionary.

## Normative References

### Requirements

ECIX II Requirements: [requirements.html](#).

### QuickData Specifications

QuickData Message Specification: [qd\\_message.doc](#).

Electronic Components Domain Specification: [qd\\_application.doc](#).

Si2 Registry Specification: [si2\\_registry.doc](#).

## ECIX Product Information Object

EDASymbol Specification:

Footprint Specification:

## Other Specifications

Electronic Components Technical Dictionary Specification: [ectd.doc](#).

SSL V3.0: <http://www17.netscape.com/eng/ssl3>

RFC 2119, “Key words for use in RFCs to Indicate Requirement Levels”:  
<http://andrew2.andrew.cmu.edu/rfc/rfc2119.html>

HTTP/1.1 rfc2068: <http://www.w3.org/Protocols/rfc2068/rfc2068>

Common Gateway Interface (CGI) Specification: <http://hoohoo.ncsa.uiuc.edu/cgi/interface.html>.

OMG Unified Modeling Language Specification, Version 1.3, June 1999: <http://www.omg.org/cgi-bin/doc?ad/99-06-08>.

## DTDs

QuickData Document Type Definition (DTD): [DTD\QD\qd.dtd](#).