

1. What is ALE?

Application Level Events (ALE) is the name of an EPCglobal software standard. The standard specifies a software interface through which client applications may interact with filtered, consolidated EPC data and related data from a variety of sources. In particular, ALE provides a convenient way for applications to read and write RFID tags, interacting with one or more RFID reader devices.

ALE provides an especially convenient programming model for application writers. Using ALE, an application makes a high-level description of what data it wants read from or written to tags, over what period of time, and with what filtering to select particular tags. The ALE implementation then finds the most appropriate way to fulfill such requests, interacting with RFID readers or other devices as necessary. Application writers are shielded from details of device configuration and management, and may rely upon the ALE implementation to handle data conversions including those specified by EPC and ISO data standards.

2. Who would use the ALE specification?

Application writers who are writing applications that have to read or write EPC data to RFID tags or other data carriers will find ALE to be a very appropriate and easy-to-use application programming interface. In general, ALE is a better starting place for writing application business logic as compared to lower-level “reader protocols,” as ALE allows the application writer to focus on data and not worry about the lower-level details of device control. Authors of applications that use ALE to work with RFID tags will find the ALE specification to be the definitive reference on how the ALE interfaces work and what behavior is standardized.

Solution providers who create implementations of ALE will find the ALE specification to be the definitive reference on what they have to implement in order to be compliant with the standard, and thus offer the services that application writers expect.

3. What kinds of products might implement ALE?

Several kinds of commercial products are available that provide ALE as a standard interface for application writers to use. These include:

- Software “middleware” applications. These are software products that typically interface with many different RFID readers, printers, and related devices via a local area network and provide for control and management of those devices. Middleware products such as these typically provide ALE as an Application Programming Interface (API) for application-specific software that is developed on top of the middleware.
- Hardware “controller” devices. These are hardware products that perform much the same function as software “middleware” does, but are delivered to the end user as a network-attached “appliance” rather than as software that is installed on a general-purpose computer. As with the “middleware” applications, “controllers” typically

Frequently Asked Questions - ALE 1.1

provide ALE as an Application Programming Interface (API) for application-specific software that is developed on top of the controller.

- “Smart” readers or printers. These are RFID readers and printers that provide a more complete application development solution than do “thin” readers or printers. By offering ALE directly as an interface, a “smart” reader or printer allows applications to be built at the same high level that ALE provides without the need for an outboard “middleware” or “controller” implementation. On the other hand, a “smart” reader ALE implementation typically only provides access to the reader in which it is embedded, while “middleware” and “controller” implementations of ALE typically allow for data aggregation or control across many reader devices.

4. Is ALE widely adopted?

ALE 1.0 was the first software specification to be ratified by EPCglobal, in early 2005. Commercial implementations of ALE 1.0 have been available since early 2004 (some of them predating the ratification of the original standard). As of February 2008, there are 18 commercial products that have received EPCglobal certification for implementing the ALE 1.0 specification. Many other products that implement ALE are known to exist.

ALE 1.1 has just been ratified, but it is expected to enjoy as much if not a greater level of adoption.

5. Is there a certification program for ALE?

As noted above, EPCglobal has certified 18 commercial products as compliant with the ALE 1.0 specification as of February 2008.

ALE 1.1 has a similar certification program to ALE 1.0. More information can be found at the [general EPCglobal Certification Programs page](http://www.epcglobalinc.org/certification/). See: <http://www.epcglobalinc.org/certification/>

6. How does ALE relate to EPCIS?

The Electronic Product Code Information Services (EPCIS) specification defines a data language for representing visibility information, namely events having four dimensions of “what,” “when,” “where,” and “why.” The EPCIS specification also provides standard interfaces for capture and query of events.

Visibility information at the EPCIS level is often used to record what took place in an operational business process that involves the handling of physical assets, such as the receipt of goods through an entry door of a warehouse. A software application responsible for supervising such a process and generating EPCIS data is called an “EPCIS Capturing Application.”

To the extent that an EPCIS Capturing Application interacts with EPC data and/or RFID tags in the course of carrying out its function, it may use ALE as the way to read or write those EPC data and/or RFID tags. In some cases, the data received from the ALE interface by the EPCIS Capturing Application may translate very directly into EPCIS data, as when EPCs are read passively by a stationary reader as goods move through a

Frequently Asked Questions - ALE 1.1

fixed doorway with no validation. In other cases, an EPCIS Capturing Application may be responsible for a complex orchestration of RFID devices, material handling equipment, and human tasks that are involved in carrying out a business process. In such cases, ALE is used specifically to interact with the RFID devices, and is therefore a smaller part of the whole.

Looking at it another way, primarily ALE answers 'What', 'Where' and 'When'. EPCIS adds the 'Why' (i.e., the business context). For example, the 'Where' in ALE usually a logical reader name. It is converted to a business location in EPCIS.

Further, the ALE interface is exclusively oriented towards real-time processing of EPC data, with no persistent storage of EPC data required by the interface. Business applications that manipulate EPCIS data, in contrast, typically deal explicitly with historical data and hence are inherently persistent in nature.

Architecturally, the ALE layer is concerned with dealing with the mechanics of data gathering, and of filtering down to meaningful events that are a suitable starting point for interpretation by business logic. Business layers, where EPCIS comes into play, are concerned with business process and recording events that can serve as the basis for a wide variety of enterprise-level information processing tasks.

7. How does ALE relate to reader protocols such as the EPCglobal Low-Level Reader Protocol (LLRP) and the EPCglobal Reader Protocol (RP)?

Generally speaking, ALE exists at a higher level of abstraction than reader protocols. An implementation of the ALE APIs may use the reader protocols to communicate with individual reader devices.

The difference in abstraction level shows up in several ways:

- In ALE, the emphasis is on providing the most convenient programming model for application writers. Using ALE, an application makes a high-level description of what data it wants read from or written to tags, over what period of time, and with what filtering to select particular tags. The ALE implementation then finds the most appropriate way to fulfill such requests, interacting with RFID readers or other devices as necessary. In contrast, the emphasis in a reader protocol is on providing complete visibility and control over the operation of a reader.
- In ALE, an application may use logical reader names to decouple the identity of individual reader devices from the names used by applications to refer to data sources. A reader device can be replaced, or even substituted by a different number of readers, without the knowledge of an ALE client application. In contrast, use of a reader protocol requires addressing a specific reader device.
- In ALE, application writers are shielded from details of device configuration and management. Reader protocols, in contrast, are designed to provide control and management of reader configuration.

Frequently Asked Questions - ALE 1.1

- ALE provides data conversions including those specified by EPC Tag Data Standards and ISO 15962. Reader protocols operate at the level of the binary data found on the tag.
- The ALE interface allows multiple readers to be controlled simultaneously. A reader protocol, in contrast, provides control of a single device.
- The ALE interface allows several applications to share data from the same readers simultaneously, without any coordination between those applications. The ALE implementation is responsible for recognizing when different clients attempt to share the same readers, and then operating those readers in such a way as to satisfy all clients simultaneously. In contrast, the LLRP Reader protocol only allows access by one client at a time. The RP Reader protocol does permit implementations to support multiple clients, but because they share the same underlying reader state those clients typically need to coordinate to avoid interfering with each other.

8. How is ALE v1.1 different from ALE v1.0?

The ALE 1.1 specification introduces several extensions and enhancements of ALE 1.0 features. New features include a significantly enhanced API for reading tags, plus four new APIs for writing tags, defining named memory fields on tags, defining logical readers, and controlling access to the ALE APIs. ALE 1.1 provides full access to all features of EPCglobal UHF Class 1 Gen 2 tags including all four memory banks and operations such as lock and kill. ALE 1.1 also includes built-in support for encoding and decoding memory according to ISO 15962.

ALE 1.1 Interfaces

- Reading API: Through this interface clients may obtain filtered, consolidated EPC and other data from a variety of sources. In particular, clients may read RFID tags using RFID readers. The ALE 1.1 Reading API is a backward- and forward-compatible extension to the ALE 1.0 API.
- Writing API: Through this interface clients may perform operation upon EPC data carriers through a variety of actuators. In particular, clients may write RFID tags using RFID “readers” (capable of writing tags).
- Tag Memory Specification API: Through this interface clients may define symbolic names that refer to data fields of tags.
- Logical Reader Configuration API: Through this interface clients may define logical reader names for use with the Reading API and the Writing API, each of which maps to one or more sources/actuators provided by the implementation.
- Access Control API: Through this interface clients may define the access rights of other clients to use the facilities provided by the other APIs.

Full Gen2 Support

The ALE 1.1 specification is designed to provide full access to the functionality of the EPCglobal UHF Class 1 Gen 2 Air Interface (“Gen2”, also known as ISO 18000-6C) specification, when interacting with Gen2 / ISO 18000-6C RFID tags. This includes reading and writing all memory banks, as well as exercising specific operations such as “lock” and “kill.”

Support for ISO 15962

ALE 1.1 introduces new features to provide built-in support for encoding and decoding tag memory according to ISO 15962. ALE 1.1 defines variable fieldnames (of the form *@bank.oid*) and interprets them as ISO 15962 “data sets” (*oid* refers to Object Identifiers of the “data sets”) for reading and writing in data carriers following ISO 15962 data encoding rules. Moreover, there are operations defined to initialize and validate ISO 15962 encoded memory.

9. Is the new ALE interface backward and/or forward compatible?

The ALE 1.1 specification is fully backward compatible with ALE 1.0. Backward compatibility means that an ALE 1.1 implementation can serve ALE 1.0 clients. Some features of ALE 1.0 are deprecated as they are subsumed by new features in ALE 1.1, but the old features continue to be supported for ALE 1.0 compatibility.

Likewise, ALE 1.0 is generally forward compatible with ALE 1.1. Forward compatibility means that an ALE 1.0 implementation can be used by an ALE 1.1 client. However, there are new features in the Reading API that ALE 1.0 implementations do not support (e.g. *startTriggerList*). Such features will generally be ignored by an ALE 1.0 implementation if an ALE 1.1 client attempts to use them. A list of the changes to the Reading API are provided in Section 16.1 of the ALE 1.1 Specification, Part 1. A client can determine the version of the ALE Reading API implementation by using the *getStandardVersion* method.

ALE 1.1 is designed so that backward and forward compatibility will continue to be maintained in future versions of the ALE specification.

10. We started using Gen2 tags commercially. Does ALE 1.1 support Gen2?

Yes. In fact, most implementations of ALE 1.0 also support Gen2, insofar as they are capable of reading Gen2 tags. What ALE 1.1 provides that ALE 1.0 did not is full access to all features of Gen2 tags, including the ability to read and write all four memory banks and carry out Gen2-specific operations such as lock and kill. Please see the answer to Question 1, above.

11. We have not adopted Gen2 yet. Can we use ALE 1.1?

In short, the answer is yes. Data carrier specific details are made fully transparent to ALE 1.1 client. It is the responsibility of the ALE 1.1 implementation, to interpret (and document as well) different fieldnames, formats and operations when operating with tags other than Gen2.

ALE was primarily conceived and developed in the context of RFID tags. However, the interface is designed to be general enough to accommodate other kinds of data carriers, such as bar codes, OCR text, and in some instances even human interaction through a keyboard or display. In the context of ALE, *Tag* refers to RFID tags or any other data carrier that can be treated in a similar manner, and *Reader* is used to refer to a channel through which Tags are accessed.

12. We still use bar code technology for inventory management. Can we adopt ALE 1.1?

Please see the answer to Question 11, above.

13. What are *Event Cycles* and *Command Cycles*?

An *event cycle* or *command cycle* is an interval of time over which an ALE implementation carries out interactions with one or more *Readers* on behalf of an ALE client.

An *event cycle* is the smallest unit of interaction between an ALE client and an ALE implementation through the ALE Reading API. An *event cycle* is an interval of time during which *Tags* are read in the Reading API. A *command cycle* is the smallest unit of interaction between an ALE client and an ALE implementation through the ALE Writing API. A *command cycle* is an interval of time during which *Tags* are written, or other operations performed upon them, in the Writing API.

An ALE client sends an *ECSpec* to an ALE implementation to describe an *event cycle* and one or more reports (*ECReport*) that are to be generated from it. It contains a list of logical Readers whose data are to be included in the *event cycle*, a specification of how the time boundaries (start and stop conditions) of *event cycles* are to be determined, and a list of specifications each of which describes a report to be generated from this *event cycle* (that is, what data are to be read and how they are to be reported).

Similarly, An ALE client sends an *CCSpec* to an ALE implementation to describe a *command cycle* and one or more reports (*CCReport*) that are to be generated from it. A *CCSpec* contains a list of logical Readers whose tags are to be operated upon in the *command cycle*, a specification of how the time boundaries (start and stop conditions) of *command cycles* are to be determined, and an ordered list of sequences of commands to apply to *Tags* in this *Command Cycle*.

14. What are *ECSpecs* and *ECReports*? What are *CCSpecs* and *CCReports*?

Please see the answer to Question 13, above.

15. I want the communication between ALE client and ALE implementation to be secure. Does the protocol support secure communication?

Both, synchronous and asynchronous (callback) communication between ALE 1.1 clients and ALE implementations can be secured using HTTPS (HTTP over TLS) in ALE 1.1. However, the ALE 1.1 specification leaves it up to the ALE implementation if HTTPS is supported.

16. How is Access Control supported in ALE 1.1?

The Access Control API is a new feature introduced in ALE 1.1. Using this API, an administrator (a sufficiently privileged ALE client) can control access to ALE APIs by other ALE clients. Clients can be granted access to all ALE features, only to specific

Frequently Asked Questions - ALE 1.1

ALE APIs, or even to specific methods of specific APIs. The Access Control API is designed to support role-based access control, and to allow implementations to leverage external identity management systems such as LDAP directories.

17. When Access Control is used, how is the identity of an ALE client authenticated?

Each binding of the ALE interfaces may provide mechanisms for authentication of client identities. For example, HTTP basic authentication and TLS certificate-based authentication are both available when ALE is carried over a SOAP/TLS/HTTP stack. How a client identity is authenticated in any particular implementation is outside the scope of the ALE 1.1 specification.

18. Does ALE 1.1 allow a user to define his own logical readers?

Yes. The Logical Reader API enables the user, in a standardized way, to define a new logical reader name as an alias for one or more other logical reader names. In the context of ALE 1.1, *composite reader* refers to a logical reader name that has been defined as an alias for one or more other logical reader names, and the term *base reader* refers to a logical reader name that is not defined as an alias, and instead corresponds to an actual channel for manipulating Tags. However, the ALE 1.1 specification leaves it up to the implementation to provide means for configuring *base readers*.

19. May I extend ALE 1.1 with implementation specific customizations?

Vendor specific attributes and elements can be added to XML binding for ALE 1.1. However, the vendor extensions must comply with some rules to conform to ALE 1.1 standard (See section 3.2, part 2 of ALE 1.1 specification). These rules ensure full forward- and backward-compatibility. In addition, vendor specific methods can also be added to any ALE API.

20. ISO 15961/15962 has been adopted in a variety of industries. How does ALE 1.1 embrace this standard?

Please see the answer to Question 8, above.

21. How does ALE 1.1 support tag commissioning?

The introduction of the Writing API in ALE 1.1 provides users with a highly flexible and full-featured mechanism for commissioning tags. By defining Command Cycle Specifications, clients may choose which tags to operate on as well as specify the operations to perform and the order thereof. Operation types include reading from memory, memory consistency checks, memory initialization, adding new fields, writing to existing fields, deleting fields, providing access passwords, killing tags, and locking fields. This allows all of the tag operations normally required during commissioning to be carried out.

A common requirement in tag commissioning use cases is the need to assign a different EPC for each tag that is written. The ALE 1.1 Writing API includes a feature called an “EPC Cache” that facilitates this. A client may initialize an EPC Cache with a list of EPCs, either enumerated individually or provided via a range of serial numbers. Thereafter, a new EPC can be assigned and written to each tag without any further intervention by the client. Each report resulting from the writing of a new tag will indicate the specific EPC that was assigned from the EPC Cache.

Another feature of the Writing API allows the value written to a given field of a tag to be generated by a Random Number Generator. This supports commissioning use cases where a unique access or kill password is to be assigned to each tag.

See the ALE 1.1 Specification, Part I, Section 12.2, for example use cases of this kind.

22. How does ALE 1.1 support tag killing?

The Writing API in ALE 1.1 provides the ability to perform all available operations on Gen2 tags, including the kill command. Through the Writing API, the client may direct a reader to kill a tag, using a specified kill password.

A common requirement in tag killing use cases is the need to select a kill password that is different for each EPC. The ALE 1.1 Writing API includes a feature called an “Association Table” that facilitates this. An Association Table is a lookup table that is indexed by EPC. A client may initialize an Association Table so that a corresponding kill password may be looked up for each EPC. Thereafter, tags that match the selection criteria may be killed without any further intervention by the client. Each report resulting from the killing of a tag will indicate the specific EPC that was detected and killed.

See the ALE 1.1 Specification, Part I, Section 12.2, for example use cases of this kind.

23. Is ALE a “binary” protocol?

The wire-level protocol for ALE is based on XML and SOAP technology, specified using XSD and WSDL description languages, which makes ALE very easy to integrate into a large number of application development platforms and tools. ALE does not provide a “binary” protocol in the sense that the EPCglobal Low-Level Reader Protocol (LLRP) does.

In the ALE specification, ALE is defined abstractly using the Unified Modeling Language (UML), and then separately a concrete binding to XML and SOAP syntax is provided. It is technically feasible to construct an alternative binding based on a binary syntax similar to LLRP, although there is currently no effort underway within EPCglobal to do so. Any party interested in the possibility of a binary binding for ALE should consider joining EPCglobal to make their interest known.

For additional questions or comments, contact the EPC Admin at epcinhelp@epcglobalinc.org and your request will be forwarded to the facilitator of Filtering & Collection 1.1 Work Group.

Deleted: ¶
¶

Formatted: Left, Don't adjust space between Latin and Asian text

Formatted: Font: 11 pt