IST Amigo Project

Detailed Design of the Amigo Middleware Core Security & Privacy, Content Distribution, Data Storage

Public

Project Number	:	IST-004182
Project Title	:	Amigo
Deliverable Type	:	Report

Deliverable Number	:	D3.1c
Title of Deliverable	:	Detailed Design of the Amigo Middleware Core Security & Privacy, Content Distribution, Data Storage
Nature of Deliverable	:	Public
Internal Document Number	:	Amigo_WP31c_v1.0
Contractual Delivery Date	:	31 August 2005
Actual Delivery Date	:	13 September 2005
Contributing WPs	:	WP3
Editor	:	**Microsoft:** Ron Mevissen
Author(s)	:	**IMS:** Marco Ahler, Viktor Grinewitschus, Christian Ressel
		TID: José María Miranda, Álvaro Ramos
		Microsoft: Ron Mevissen, Stephan Tobies

Abstract: This document describes the detailed design for Security & Privacy, content distribution and data storage. The design for Security & Privacy is a refinement of the security & privacy architecture as it was devised in WP2. The refinement concentrates on service discovery for concrete protocols, the interoperability with legacy devices, and authentication and authorization processes. It also describes an early version of the biometrics security infrastructure that will be further developed in our future work.

The content delivery design uses the architectural pieces described in WP2 to present a solution for the interoperability between different kinds and formats of content and the diversity of media rendering equipment found in an automated home like Amigo. Legacy content as well as legacy devices and processes how they can be integrated into the proposed design are described. Interoperability is realized by an adaptive system that prepares content off-line for the different classes of devices.

Data storage solves the responsibility of an automated home with respect to the secure and reliable storage of all kinds of data. Inhabitants of an Amigo home will only trust the system if they can rely on the system keeping their data secure in relation to access and persistence. Secure storage differs from the general Security & Privacy concepts because the security attributes are related, and have to be stored, with the data. General Security & Privacy on the other hand deals mainly with dynamic aspects like communication access and authorization requests.

Keyword list: ambient intelligence, networked home system, mobile/personal computing/consumer electronics/home automation domain, interoperability, middleware, service discovery protocols, content distribution, multimedia streaming, security, privacy, data storage

Table of Contents

1 Introduction

An automated home, as the word indicates, automates and thereby aims to ease numerous daily tasks and routines. Although this provides a clear advantage, it also creates new fears for the inhabitants of such an environment: the impression that they lose (part of) the conventional control they had over existing and new functions. Multi-modal interfaces, pervasive, ubiquitous computing devices and wireless networks contribute to the user's impression that anything is possible and that they no longer have full control over all functionalities. Additionally, inhabitants of such an automated home often have to trust the system with a significant amount of their personal information such as profile and context information.

Key success factors for such a system are the guarantees and implemented functionalities for delivering an adequate level of security and privacy protection. For the end user, this translates into a level that is comparable to the current (often implicit) level of security. When he closes the doors of his house, nobody can enter and access any personal information or activate any functions in his house. If there are people in the house he automatically trusts them to use certain domestic functionalities, but still has ways to protect personal information (by locking it away for example). The Amigo system, however, also extends to application developers and service providers. To ensure the overall security of the system when it is being extended in the future, guidelines (WP4) and components need to be designed that allow application designers to easily embed security & privacy functionalities into their Amigo applications. This is the main topic of chapter 2.

Privacy directly relates to the integrity and confidentiality of stored data. If every application implements its own way of storing data it is nearly impossible to assess and guarantee a system-wide privacy protection level. Chapter 4 therefore introduces a generic storage solution that can be used by all components in the system. It can be used to protect data against unauthorized access (confidentiality), to share data among applications and to issue notifications when data has been changed.

Chapter 3 provides a detailed architecture for content distribution in the Amigo system. The focus here is on the interoperability of content with the different devices in an Amigo system and the available capacity of the network infrastructure. Content may arrive from many sources and in different formats. Different content rendering devices may have different requirements on the content's format. The design described in this chapter enables seamless playback of any content on any available device with regards to optimal bandwidth usage.

2 Security & Privacy

2.1 Introduction

In [D2.1], the specific security and privacy requirements that apply to the Amigo home were analyzed. Based on these requirements, a security architecture was proposed that provides the necessary privacy, as well as authentication and authorization as part of the Amigo middleware. In this document, this architecture will be refined towards an implementation of the required components.

2.1.1 Requirements

For the reader's convenience, the main requirements and proposed architecture are repeated here. See [D2.1] for a more thorough discussion and motivation of these requirements.

Interoperability
Deal with a diverse array of devices with potentially different capabilities and usage policies. It is imperative that the security and privacy Architecture takes all these categories of devices into account whilst still guaranteeing a sufficient level of security and privacy.

Pre-configured
Guarantee an initial minimal level of security and privacy settings.

User-friendly
Configuration should be possible without detailed knowledge of the underlying technologies; decisions and their implications should directly relate to the Amigo environment as seen by an Amigo user.

Self-managed
No central user; information necessary for the security system should be maintained by the system itself; interaction with the security system, when required, should be performed with as few actions as possible.

Distributed
The security system should be as resilient as possible, and, therefore, should not become a single point of failure in an Amigo home; if part of the security system fails, it should still be possible for users to continue with a limited number of services or limited functionality of these services.

Dynamic
Simple integration of new or temporary devices (rented, borrowed or carried by visitors) into an Amigo home; allow these devices to leave and re-enter the network without the need to continually re-register these devices.

2.1.2 Proposed Architecture

Document [D2.1] sketched a middleware architecture that provides mechanisms to meet these requirements. Before we refine this architecture, we will provide a brief outline of the architecture; again, see [D2.1] for more information:

The basic building blocks of the security and privacy architecture are two middleware services, namely, the authentication service and the authorization service. The authentication service handles verification of identities whilst the authorization service handles access control to resources by that identity. The authentication solution is based on Kerberos [KoNe93] which has been extended with identities for devices. The authorization process is specifically designed for the networked home system using a Role Based Access Control (RBAC) approach [RBAC].

Users, devices (if capable), and services (if they need to be secure) acquire identity tokens (also called ticket-granting tickets) from the authentication service. When accessing a secure service, both the user and the device identity tokens are presented to the authorization service to acquire a service-specific. The authorization service validates this request and, if applicable, issues a service-specific token that can be used by the device to access the service. The end-service validates the token and grants (or denies) access. As a by-product of this authorization process with Kerberos, the client and service establish a shared secret key that can be used to protect the subsequent session.

Services in the middleware and application layer can be divided into two categories: standard and security-enforcing. Security-enforcing services are like any other service, except that they (1) require clients that contact them to provide an authorization token, and (2) use a secure form of service discovery. A security enforcing service verifies the validity of the provided authorization token before will provide its services. Standard services do not impose this requirement on a client, and are hence considered unsecured.

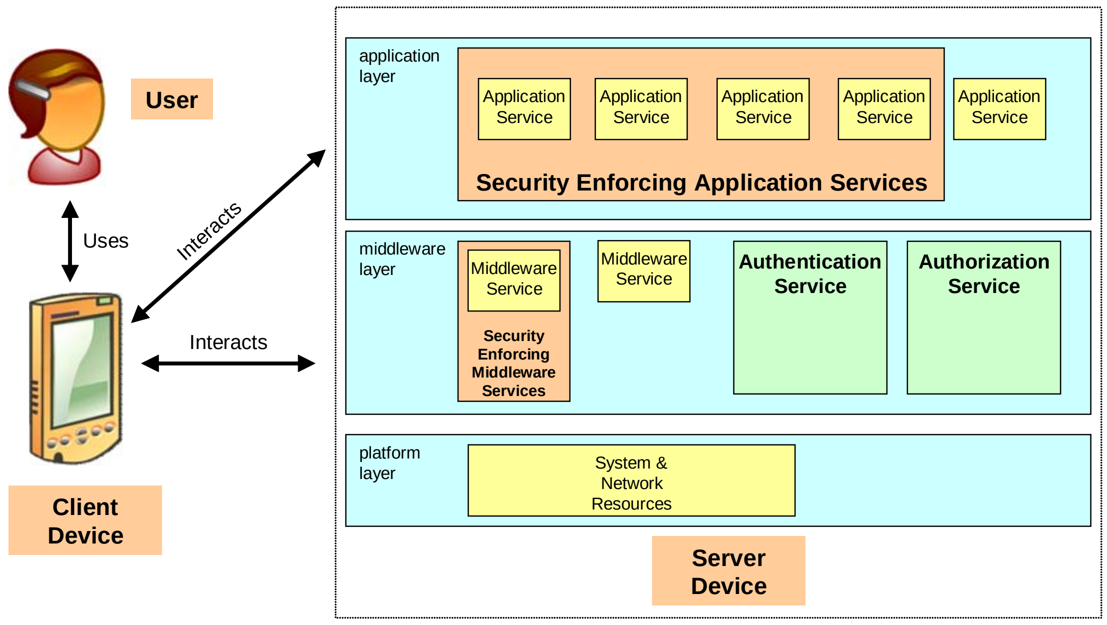


Figure 1: Security and privacy architectural components

2.2 Security and Privacy for Generic Service Discovery

Being able to query available services in a user’s home has significant privacy implications. Thus, service discovery should be restricted to authorized devices. Special care needs to be taken to develop a solution that works with the Amigo event-based approach to service discovery interoperability described in [D2.1].

2.2.1 Encryption for service discovery and its integration into the Amigo service discovery interoperability approach

Service discovery messages should be encrypted to reduce the risk of an eavesdropper acquiring that information. Knowledge of the encryption key also serves as a simple authentication mechanism between the services to control participation in service discovery.

The employed encryption scheme should work with all sorts of devices, so it should be kept as simple as possible. Thus, we use a standard symmetric encryption algorithm with a secret key (the “house key”) that is shared between all trusted devices of an Amigo home. Acquisition of the house key (how a new device initially gets to know this key) is described in Section 2.3. We assume that all devices that the user has granted unlimited access to the home have a means of acquiring knowledge of the house key.

Guest devices, i.e., devices that are only admitted to the home environment for a limited amount of time and may only be able to use a restricted number of the available services need to use a different mechanism (see 2.2.3). The same also applies for legacy devices, i.e., devices that have no support encryption in service discovery.

When an Amigo device wants to actively discover a service, it will use the house key to encrypt the service discovery request, thus proving that it is a trusted device who is allowed to participate in unrestricted service discovery within the Amigo home. Service announcements will also be sent encrypted with the house key so that they are only available to trusted devices. Any device discovery message that cannot be successfully decrypted with the house key shall be considered to be a legacy message (see 2.2.3) and will be discarded by Amigo clients and services. There is one notable exception to this behavior: the authentication and authorization services will have to act on these messages to allow legacy and guest devices entry to the Amigo home.

To allow the revocation of trust from previously trusted devices, there must a possibility to change the house key. This mechanism is described in section 2.3.3.

2.2.2 Integration of this encryption scheme into the event-based Amigo service discovery interoperability

In Amigo, interoperability between different service discovery protocols is based on a parser/composer model that communicates using events which capture the abstract nature of the service discovery messages in the concrete protocols. Integration of the proposed encryption scheme into this mechanism can easily be accomplished by adding an SDP_SECURITY_ENCRYPTED event. A parser with knowledge of the house key will attempt to decrypt incoming service discovery messages using the house key and will generate an additional SDP_SECURITY_ENCRYPTED event if this decryption was successful. Similarly, a composer that receives this event knows that it should encrypt the outgoing message with the house key before sending it. Internally, the decision to relay or drop a service discovery message can be based on the presence or absence of a SDP_SECURITY_ENCRYPTED event.

2.2.3 Treatment of legacy devices, guest devices, and newly acquired devices

There are devices that will not be able to participate in the encryption scheme for service discovery, either because they are not aware of this scheme or cannot implement it (e.g., due to a lack of resources), or because they are not sufficiently trusted to provided access to the house key (e.g., in the case of guest devices). Finally, newly acquired devices that are added to the home need to have some means of discovering the authorization and authentication service (e.g. to acquire the house key). For the purpose of service discovery, we do not need to distinguish between these classes of devices and will simply refer to them as restricted devices.

In the following sections, we will describe our approach for enabling interoperability of service discovery between restricted devices and the Amigo home. We have already indicated that most Amigo services and clients will discard service discovery messages that have not been encrypted with the house key, with the exception of the Authentication and Authorization Service (AAS) of the Amigo home.

2.2.3.1 Making Amigo services discoverable for restricted clients

To allow restricted clients to find Amigo services, the AAS can answer unencrypted service discovery messages that it receives from a restricted client. The range of services that is advertised in this manner can both be based on a classification of the services and on the trust-level that the user assigns to the restricted client. The latter will require some form of authentication of the restricted client, which could, for example be based on a restricted client's MAC address, or some other form of authentication. Also for guest devices, some Amigo-specific authentication scheme could be used.

This approach works well with both repository-based and active service discovery, where the AAS can choose the announced services based on the identity of the restricted client. It does not work well with passive service discovery, where the multicast nature of advertisement messages will make it impossible to address single restricted clients. Two possible solutions will be investigated during design: (1) the AAS decides to relay an encrypted service announcement in unencrypted form, in which case it will be visible to all clients on the network or (2) it decides not to relay it, in which case the service is effectively invisible to all but the trusted devices.

2.2.3.2 Making restricted services discoverable for Amigo clients

To allow Amigo clients to discover restricted services, we choose a similar way via the AAS. If service discovery is active, the AAS can decide if it wants to relay incoming encrypted announcements to the restricted domain or not. In this process, the identity of the original requestor can be replaced by the identity of the AAS, thus hiding the identity of the requestor from the restricted devices. Answers from the restricted devices will then be relayed back into the Amigo domain, encrypting them in process. The AAS can also choose to answer the requests directly, based upon previous knowledge of available restricted services.

If service discovery is passive, the AAS will relay service announcements from restricted devices to Amigo devices, based on the user's configuration of the restricted devices which should be made available to Amigo devices.

2.2.4 Security and Privacy for Specific Service Discovery Protocols

In the previous section, we discussed how secure and private service discovery will work in the Amigo home. The discussion was only on the level of an abstract service discovery protocol, which ensures interoperability of our solution between different service discovery protocols. In this section, we describe how this abstract approach to service discovery is realized using a number of concrete service discovery protocols, which serves to prove the feasibility of this generic approach.

2.2.4.1 WS-discovery

WS-discovery [WSD05] messages are formatted as Simple Object Access Protocol (SOAP) envelopes, which makes it easy to encrypt them in a non-proprietary way using WS-Security [WSS04].

There are several alternatives regarding which portion of the envelope shall be encrypted. Encrypting the full envelope will provide maximum privacy, but will also require that all listening devices always try to decrypt each discovery message. This is wasted effort when, for example, services are only interested in "Probe" and "Resolve" messages, but have to first decrypt each incoming message, only to discard "Hello" and "Bye" messages as they are only relevant to clients.

We propose to only encrypt the Body part of messages, which thereby conceals "interesting" confidential information about the involved clients and services. This allows discarding of some "uninteresting" messages based on the SOAP header.

We assume knowledge of a secret key that is shared between all authorized devices in the Amigo home. This key is used to encrypt the SOAP body that contains the WS-Discovery message, using a suitable specified symmetric encryption algorithm by replacing the SOAP body with the resulting <xenc:EncryptedData> element, using one of the supported encryption methods, e.g., [TripleDES] in CBC mode. The resulting message is then sent as a normal WS-Discovery message via UDP on a well-known IP-multicast-group, just like a normal WS-Discovery message.

Upon receipt of an encoded WS-discovery message, the listening client or service will attempt to decode the message using the shared key and the specified encryption method. If this results in a valid soap message, the sender of the message is considered to be authorized, and the message is processed in the normal manner. Failure to decrypt the message into a valid WS-discovery message will cause the message to be discarded as it is assumed to have originated from an un-authorized client.

2.2.4.1.1 Treatment of Legacy Devices

Non-Amigo devices that support WS-Discovery will not be aware of the employed authorization scheme and instead, will send their messages unencrypted. To allow these legacy devices to be used in the Amigo home, we refine the generic approach to deal with legacy devices described in Section 1.2.3 above.

Legacy clients will multicast unencrypted probe messages. Normal Amigo services will simply ignore these messages. Instead, the AAS will act as a WS-Discovery proxy for these devices and answer their requests. The types of services that will be exposed by the proxy is configurable by the Amigo user, but will usually only cover elementary services, whose exposure does not jeopardize the privacy of the home.

Legacy services will not be able to understand encrypted service discovery requests. Thus, they can only be discovered using unencrypted requests, which are not normally sent by Amigo clients. To make legacy services visible to Amigo clients, the AAS will act as a WS-discovery proxy and will include these legacy services into the request matching process. For this, the AAS will monitor the network for unencrypted Hello and Bye messages to maintain an accurate picture of the currently available legacy devices.

2.2.4.2 Universal Plug and Play

Universal Plug and Play (UPnP) defines a device security service that is capable of providing strong authentication, authorization, replay prevention, and privacy of UPnP SOAP actions [UPnPDS03]. Yet, no mechanism to secure the discovery protocol of UPnP (SSDP) is given. Augmenting SSDP with symmetric encryption, in a similar way that has been proposed for WS-Discovery in section 2.2.4.1 would be an appropriate solution that provides additional privacy for the Amigo Home.

2.2.4.3 Jini

Like UPnP and WS-Discovery, Jini does not have direct provisions for encrypting service discovery. The binary encoding of Jini service discovery makes its extension with an encryption scheme in a non-proprietary way impossible.

2.3 Authentication and Authorization

Existing middleware implementations have widely divergent approaches to the topic of authorization and authentication and none of them appear to directly fulfill the user requirements established by the Amigo usability studies. Moreover, none of them supports the uniform architecture that has been defined in previous releases of the Amigo middleware architecture document. Thus, we propose a solution that works independently from any particular security infrastructure provided by the supported middleware and instead, is based purely on service invocations. This approach makes it possible to be integrated into the Amigo home without the need to modify any existing middleware structure – although, of course, existing services running on this middleware will have to be modified to be capable of participating in the Amigo security scheme. Specific Amigo middleware services may provide interoperability between existing middleware-level security and Amigo security.

In the following sections, we outline the protocol that is used by the Amigo security scheme. Any reader that is familiar with the Kerberos security system [KoNe93] will immediately notice the close correspondence between Kerberos and the Amigo security scheme. This is, of course, intentional. Instead of inventing our own security scheme, which would require a very thorough analysis to ensure its security, we start from a proven base, and carefully adapt it to better serve the needs of Amigo. This approach enables simpler analysis of the resulting protocol, based on existing Kerberos knowledge.

2.3.1 Authentication Service

At the center of the middleware infrastructure resides the trusted authentication service that enables participants of the Amigo infrastructure to establish and prove their identity. For this, each participant must be registered with the authentication service. During this registration, an identifier (user name, service endpoint, etc.) is created that refers unambiguously to the participant. Secondly, a shared secret, which is only known to the participant and the security service, is established. Following the Kerberos principle of authentication, this secret is then subsequently used amongst participants of the authentication service to establish mutual authentication.

2.3.1.1 Registration of a new participant

When a new device, service, client, or user wants to use the Amigo system, it first needs to be registered with the authentication service, at a well-known or publicly discoverable service endpoint. The participant will send a registration request to this endpoint, specifying their kind (service, client, device or user), requested and required access privileges, and the secret key to be used for subsequent authentication. In case of a user registration, a username could be specified, whereas the registration request for a non-human participant is accompanied by some information about the nature and construction of the device, client or service.

Depending on the nature of the registration request, an administrative user decision is required to allow the requesting participant into the Amigo home, while in some other cases simple proof of having legitimately entered the physical home might be sufficient to have registration granted (e.g., when requesting only limited guest access and location services can validate that the user/device is physically within the home).

A confirmation, together with a unique handle for the new participant is sent when the registration is granted. In case of a rejected registration, an indication of this circumstance is also sent.

2.3.1.1.1 Securing the registration process

Although this registration process is conceptually simple, a number of aspects require further consideration:

1. Initial participant authentication - how do we ensure that the participant that is being admitted into the home is the participant to whom we intended to grant access?

2. Initial AAS authentication – how do we ensure that we are registering with our home's AAS and not some other service that will then be able to abuse a newly registered service?

3. Secure communication – how do we ensure that the secret that is used for the mutual authentication of the participant and the AAS is not captured by an attacker? This knowledge could be used to impersonate a participant or the AAS itself.

We propose a simple approach that solves all three problems: controlled sharing of a secret out of band using a channel that is outside of the normal registration process. ‘Out-of-band' here refers to any means of communication that is inaccessible to an attacker, i.e., does not use the normal network resources. ‘Controlled’ refers to the fact that the secret has been successfully communicated between the device and the security service.

This solves the indicated problems: 1) and 2) are solved because the controlled propagation of the secret from the requestor of the registration to the AAS fixes both the identity of the requestor (which is the source of the secret) and the AAS (which is the sink of the secret). It also solves 3) because the secret can be used to encrypt the registration process so that an attacker cannot learn the secret shared between participant and AAS.

As an example for this principle, a newly to-be-registered device could generate a random password, which is displayed on the device’s display and needs to be entered into a confirmation dialog of the AAS. A key derived from this password is then used to encrypt the registration request. This ensures that the user exactly confirms the registration request that has been generated by the device, because no other registration request can be decrypted with this key.

An alternative means of transporting this key could be signaling of the key via barcodes read in by a camera, via infrared, RFID, or via playing sounds into a microphone. This allows for the easy integration of limited and/or headless devices into the home. Also, this mechanism could be used to allow registration for limited access without administrative user interaction if the location of the camera, infrared of RFID transceiver, or microphone would ensure that the device has already been brought into the home.

2.3.1.2 Authentication

Once a participant has been registered with the AAS, it can use the secret that it shares with the AAS to gain a so called ticket-granting ticket (TGT), which proves its identity to the authorization service. The steps required to acquire a TGT are as follows:

- The participant sends its name to the AAS – this communication does not need to be encrypted.
- The AAS checks if a participant under this name has previously been registered and generates a TGT with an expiry date according to a configured policy (but no longer than the duration of the granted registration period):
    - This ticket is encrypted with a secret key that is only known to the AAS. The ticket contains, among other information, a fresh session key that will be used to bind the TGT to the participant to which it is issued.
    - This ticket also contains a session key (and some other information like expiry) that is separately sent to the participant, encrypted with the participant’s key. Thus, it is ensured that only the participant will be able to use the TGT, which establishes the identity of the participant – an attacker or imposer that receives the TGT will not be able to determine the session key because it has only been visible in encrypted form: either with the authorization service’s key or with the participant’s key. Without the knowledge of any of these keys, the ticket cannot be used towards the AAS to gain access to secured services.

This cryptographically secured channel between participant and authentication service can also be used to convey the house key used for service discovery in a way that prevents attackers or participants with limited privileges from gaining knowledge of this key. Thus, successful authentication allows a device to participate in service discovery.

The use of TGTs in the security scheme is motivated by the necessity to reduce the amount of data that is encrypted with the key that is shared between AAS and the participant. The purpose is to reduce the risk of known-ciphertext attacks.

2.3.1.3 Authorization

Once a participant has acquired a TGT, this ticket can be used to obtain a ticket for a particular service. For this, a ticket request is sent to the AAS. This ticket request consists of the TGT, a request authenticator, and the name of the service for which the ticket shall be obtained. Upon reception of a ticket request, the AAS tries to decrypt it with its secret key, thus establishing the genuineness of the TGT. From the TGT, the authentication session key is extracted. This session key has also be used by the participant to encrypt the authenticator, thus ensuring that the authorization request has been sent by the participant to which the TGT had been issued (a timestamp embedded into the authenticator is used to guard against replay attacks). The TGT carries the identity of the participant as it has been established during authentication, thus binding the authorization request to its genuine originator.

Based on the configured roles for the service and the requestor, the AAS may or may not grant a ticket to the requestor – the specific rules for this are determined by the role-based access control scheme that has been presented in [D2.1]. If a ticket is granted, it again contains a fresh session key that is encrypted with the secret key shared between the AAS and the service for which the ticket has been generated. The ticket will also contain an expiry date, after which the ticket shall be considered invalid by the service (in which case the requester has to repeat the procedure to acquire a new ticket).

2.3.1.3.1 Logging into a service

The process to log into a service with a ticket is similar to the process of authorization above, but takes place between client and service without communication with the AAS. The client sends the ticket, accompanied by an encrypted authenticator, to the service, which is then checked by the service by first attempting to decrypt the ticket with the key it shares with the AAS. Successful decryption proves that the ticket has been issued for the service by the AAS. It also informs the service of the identity of the client, which is stored in encrypted form in the ticket and thus cannot be manipulated by the client. Once this has been established and the ticket has been found to be within its validity period, the session key from the ticket is used to determine validity of the authenticator, thus proving that the current request originates from the owner of the ticket. Within the authenticator, an additional session key and session identifier may be sent from the client to the service, thus enabling the use one ticket for a number of separately named and secured sessions. Also, a recent time stamp is included into the authenticator to guard against replays. As a result of a successful login, the service re-encrypts the timestamp from the authenticator and sends it back to the client, thus proving that it has been able to access the authenticator, which is only possible via knowledge of the session key, which in turn requires knowledge of the key with which the ticket had been encrypted. Thus, the client is sure of the identity of the service – mutual authentication has been achieved.

Note that the whole exchange does not involve any communication with the AAS – a client is able to log into a service as long as its ticket for the service is valid.

2.3.2 Securing the session

Once a client has logged into a service, service and client has mutually established proven their identities, and share a secret key liked to a session id. This secret key is known only to the client and the service and can thus be used to secure future exchanges between client and server. Depending on the security requirements, this key can be used to encrypt all subsequent communication within this session, or it can be used to accompany invocations with small proofs of authorization. The specifics of these security measures is beyond the scope of our discussion of the AAS, although releases of the Amigo middleware will provide standard, interoperable ways for securing client-service communication based on the secret key established during authorization and authentication.

2.3.3 Revoking Authorization

There are situations when a user wishes to revoke the trust from a device that has previously been added to the Amigo home, for example when a device has been sold and should no longer have access to the home.

Given the centralized nature of the Amigo security scheme, this is easily accomplished by instructing the AAS to no longer grant tickets and ticket-granting tickets to a specific device, user, or client. This will ensure that, once all tickets owned by the participant have expired, it will no longer be able to gain access to secured services.

As a second step, the house key used to secure service discovery will need to be changed to exclude the revoked participant from being able to perform or monitor service discovery. As sketched above, this house key is transmitted to the participant during successful authentication. Several approaches are possible to trigger re-authentication of devices and thus distributing the new key to all devices that shall be allowed to participate in service discovery. Probably the simplest is multicast of a message via the normal service discovery channels indicating that the current house key has been invalidated. This causes all listening participants to re-authenticate, thus acquiring the new house key. Some care has to be taken to ensure that such a re-authentication cannot be triggered by an attacker (which could then easily render the Amigo home useless by causing constant re-authentication – a classical denial of service attack). This can be accomplished by requiring the re-authentication request to be encrypted with the house key being invalidated. This prevents non-authorized devices from triggering re-authentication.

In the case of repository-based service discovery, the service repository can be instructed to no longer service requests with the old house key and, instead, instruct the requestor to re-authenticate first.

2.4 Authentication and Authorization refinement for existing middleware

As indicated, the Amigo security scheme is not intended to be implemented as an extension of existing middleware. Instead, the Amigo security scheme is intended to run on top of existing middleware, as a collection of service endpoints provided by specific Amigo middleware components (like the AAS) and by any service that implements Amigo security. Thus, no adjustment of existing middleware is necessary, other than specifying the way in which the abstract security protocol outlined above is mapped to specific services running over the existing middleware. Yet, this is an obvious step that does not need to be further elaborated in this document.

2.4.1 Interoperability Considerations

We have designed the Amigo security approach to operate on top of existing middleware, so at first glance; interoperability among participants based on different middleware comes for free by the middleware interoperability provided by the Amigo framework. Unfortunately, this is only true as long as all information that needs to be converted between different middleware is accessible by the interoperability component. This is not the case where data is sent in an encrypted form, unless the decryption key is available. This implies that the Amigo middleware interoperability will have to take encrypted data into account (e.g. by using the house key to decrypt the data, convert it into another protocol and decrypt it again).

2.5 Biometrics infrastructure

An aim of the Amigo platform is to allow the integration of biometric infrastructure to support user authentication. Like any other participant, users authenticate with the Amigo security system via the knowledge of a secret that is shared between the user and the AAS – usually a password. Knowledge of this password is required to use the ticket granting tickets issued by the AAS, and thus to gain access to services in the home.

This allows the following approach for the seamless integration of biometric authentication into the Amigo home: a trusted biometric authentication service provides a well known or discoverable Amigo-secured service endpoint. When a client wants to act on behalf of a user and thus requires the user's password, it will first obtain a TGT from the AAS. Note that this TGT can be acquired without knowledge of the password, but that the password is required to use the ticket: the necessary data that accompanies this ticket is encrypted with the user's password and is as such initially useless to the client. To use the TGT of the user, the client sends this data to the biometric service, with the request to decrypt it using the user's key. The biometric service will only grant this request once it has successfully authenticated the user and has ensured the users consent to let the client act on behalf of the user.

The biometric service is running as a secure Amigo service, thus ensuring mutual authentication between service and client, and being able to establish a secure communication channel to pass back the unencrypted data to the client.

Note that, at no point in this exchange, the user's password is revealed to the client. This ensures that then client will only be able to work on behalf of the user during the validity of the TGT.

2.6 Conclusion

We described the architecture of the Amigo security service, based on an authentication and authorization service using a Kerberos-like protocol based on shared keys and symmetric encryption. We described the interactions involved in registering a new participant and authenticating a registered participant, along with how an authenticated participant can acquire authorization to secured services. The necessary services will be implemented as middleware services on top of the amigo middleware core infrastructure, which implies that amigo middleware core services have to take secured (encrypted) data into account. We also studied how we can ensure the user's privacy during service discovery, and how trust can be revoked from a device that is removed from an Amigo home.

3 Content Distribution

3.1 Introduction

Content in an Amigo home will be available from a multitude of sources and at the same time various devices will be available for rendering this content. This scenario however introduces several problems:

- Where is the content located and how is it discovered?
- As content comes from different sources, how can it be rendered on all these devices?

One possible approach would be to require content (format and distribution) to be standardized (and enforced) and to store the content on a central server so it can be easily located. This solution however is not practical regarding interoperability with existing equipment and content, besides, the central server for content would imply a single point of failure in the architecture and, due to its functionality, impose high requirements regarding performance and reliability.

The approach proposed in this chapter is more flexible and allows content to be provided from different sources and in different formats. A schematic overview of the proposed architecture and process flow can be found below.

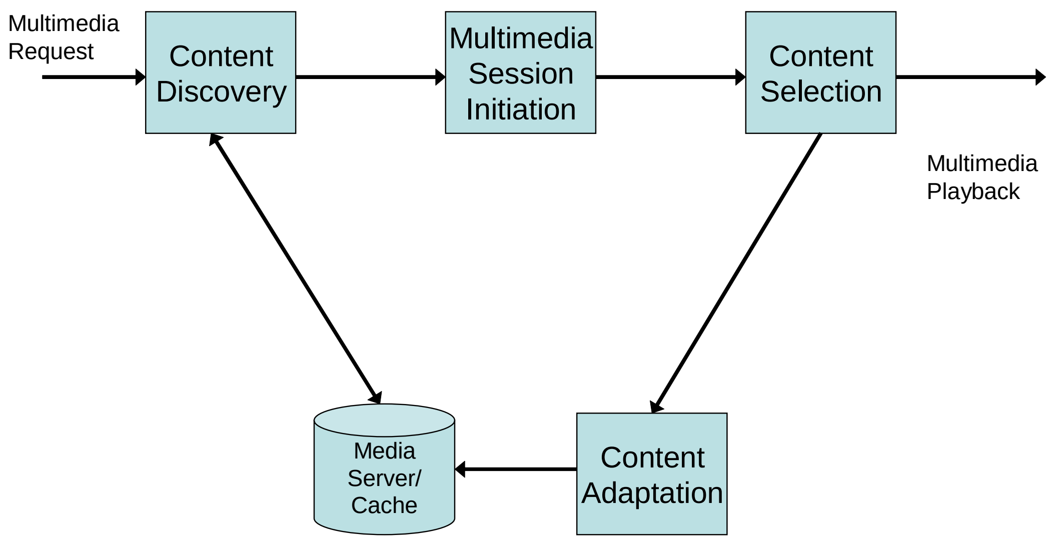


Figure 2: Content distribution process flow

A multimedia request is passed on to the content discovery component. This component locates the sources that can supply the requested media. The request is forwarded to the multimedia session initiation component (this is a legacy component, usually some kind of a controller). This component decides where (on which device) the multimedia is to be renderered. This choice is forwarded to the content selection component that uses the renderer's capabilities to select the optimal stream for that content. In case the content is not optimal for the device, the content selection component sends the content's requirements to the content adaptation module. This module will then transcode the content into the required format. Since the content adaptation module has the content in a new format it will reuse it when possible (that is, when the same adaptation is required for the same content).

The next chapters will in detail explain the design for each component.

3.2 Content Discovery

The user must be capable of seamlessly browsing or searching for content within Amigo independent of the actual device that stores the content and the actual device that runs the application. This implies that the middleware must implement capabilities that assure content discovery within the set of active content servers present in the home. In order to enable browsing and searching, two issues stand as obvious: some kind of content description is necessary to distinguish between different content; some kind of content publication is necessary to make this description available to the whole of the Amigo environment. Generally speaking, the richer this description is, the easier it will be for the user or the software to identify content. However, there is then a related higher risk of the user becoming saturated with content and the network becoming overloaded, so this needs to be considered by the design.

3.2.1 Content description

Adapting, accessing and storing content are some of the big issues to handle within the scope of Content Distribution. In order to handle these issues in the same way that Amigo tries to improve user interaction with the multimedia content in the home as well as for the content selection process (see 3.3), it is necessary to have a complete and well defined scheme for describing this content. This content description will help and serve as a basis for most of the services related to content in Amigo. Having a complete framework and language for defining content is the only way for processes like content adaptation, selection or storage to use a coherent representation and an effective way for implementing these responsibilities.

The content description proposed in Amigo is based on the work carried about in [D3.1a] about building ontology's for defining content resources, features and formats. The content description will use OWL to define a language that describes the features and formats of the different kind of content present in the home. This description will be mainly formalized, in the sense of having very well defined characteristics (format, bit rate, size…) that will be necessary for the content adaptation, selection and discovery process. In addition, some "low level semantic" description (type of content, genre, author…) will be added to the description. Furthermore, the OWL descriptions from D3.1a in the content ontology, will serve as a basis for a common description language for all the content stored in an Amigo home.

The next problem is how to use (legacy) content before it is described in Amigo. To handle this issue we will distinguish amongst three types of content:

- Legacy Content, which has no description associated with it, but is simply a binary file containing the content.
- Intermediate Content, which has been annotated using a description language (MPEG-7, DIDL-Lite...) which is different from the Amigo one.
- Amigo Content, which has already been mapped onto the content ontology and has its features annotated for Amigo purposes.

Both legacy and intermediate content also need to be discovered by users to take advantage from the enhanced Amigo features. This process is described in the next section and will consist of a framework and interoperability methods for handling those different types of content.

3.2.1.1 Content Description Interoperability

The content description interoperability problem has been posed in the section above. The proposed solution that enables interoperability consists of a common content description based on simple content semantics, representing main content semantic description features and format description. The semantic modeling of content is based on OWL and its vocabulary is presented in [D3.1a].

All three classes of content (legacy, intermediate and Amigo) may be present in the Amigo home, however each class will be treated different by the home system. Amigo content descriptions are available to Amigo-aware services and applications. Availability of Intermediate content is achieved by parsing its description and mapping it to the content ontology. On the other hand, Legacy content requires a coding specific metadata extractor or an analysis tool to obtain some description (sampling frequency of a WAV file) that will be then mapped to the content ontology.

Using a set of parsers and plug-ins, as well as an API to the ontology (e.g. JENA), intermediate and legacy content can be added to the ontology, and from there become available to Amigo-aware entities. The middleware will provide a registration service and a framework for simple installation and registration of new plug-ins or parsers. The available set of plug-ins performing description mapping to the content ontology would be accessible as a service or a set of services to other Amigo-aware entities in the home network. The architecture is depicted in Figure 3.

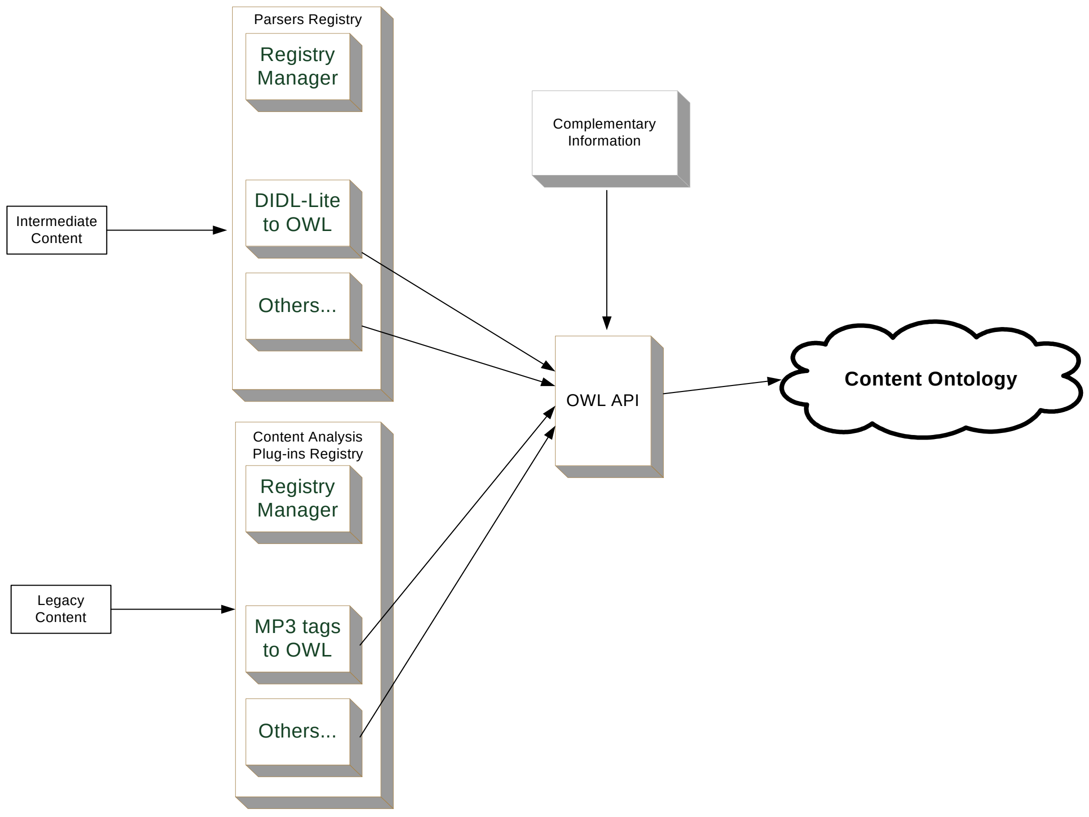


Figure 3: Content meta-data extraction mechanism

Since descriptions provided by analysis tools and parsers may be incomplete (specially for legacy content) a complementary synchronous or asynchronous input to the content ontology must be possible; this would include possible queries to the user, comments from the user, or any complementary sources for completing description (e.g. external databases such as CDDBs).

3.2.2 Content publication

Content will be stored in facilities that can be located in different devices within the home or even outside the home. This content should be available in the home network so that remote entities can find it and have access to it. In order to do so, a kind of registration and lookup service must be available in the home network. Furthermore, content may be duplicated for backup or adaptation reasons and multiple instances of the same content object can be present inside and outside the home. As in most registration and lookup solutions (such as service discovery), this can be achieved via a *centralized*, *peer-to-peer* or a *distributed* solution. This solution can also be *active* or *passive* depending on its role: the registry service may constantly look for and discover content provider entities and browse them to retrieve content metadata, or may wait for these entities to look for it and provide the information required for registration.

Registration services will take the content description as an input and would be capable of performing the description mapping to a simple common representation (see previous section) to identify content presented in different description schemes. Together with the simple description, information on the procedure for obtaining both the content resources and its original meta-data is also organized and stored. DLNA guidelines establish UPnP AV as the discovery and interaction protocol for content distribution services. However, other instances of these services may be accessed using any other protocol (i.e. Web Services, RMI) exploiting middleware interoperability mechanisms. Therefore, the Content Discovery Service should be flexible enough to allow interaction with the content distribution service in the home network using different access protocols.

Content may be duplicated for backup or adaptation reasons and multiple instances of the same content object can be present inside and outside the home. MPEG-7 solves this issue by defining *"the Media Information DS which identifies the master media, that is, the original source from which different instances of the AV content are produced. The instances of the AV content are referred to as Media Profiles, which are versions of the master obtained perhaps by using different encodings, or storage and delivery formats"* [JMM04]. However, multiple content description schemes may be present in the home, so a higher level technique would be necessary to map different descriptions into a unique representation in order to identify the master object and the distributed different instances.

In terms of efficiency it is clear that a passive registry is superior: traffic and processing load is only generated when necessary: content registration or elimination. However, in terms of robustness an active registry service can detect more easily consistency failures which can be due, for example, to an application termination after deleting some content and before updating the registry. A dual registry is more versatile, and it compromises between robustness and efficiency, depending on the weight assigned to activeness and passiveness respectively. Such a registry supports registration services for registry-aware entities that introduce content into the home and active discovery and registration for the rest, which will be henceforth addressed as intermediate directory services.

In terms of content discovery, such a dual registry would:

- Provide registration services that would receive descriptions and locations of content resources as input.
- Support description translator tools so that description mapping to a simple common representation is possible (see previous section).
- Implement active content discovery based in plug-in robots.

Registration services will produce a logical representation of the content available in the network which will be complemented by the information produced by the plug-in robots which will navigate through passive servers and register if possible to content update events. The need for these plug-ins is clear when presenting a home network based on DLNA directives: devices providing content would be UPnP Media Servers which provide a ContentDirectory service to access content meta-data and resource description. Some directory services may only provide incremental navigation through descriptions that may be organized in containers forming a tree structure; information extraction in that case would require some recursive processing that can be easily implemented by a robot.

The Content Registry services will be able to provide a logical representation of the content resources including a description of the content object and the available formatted instances with their corresponding format descriptions and locations. The latter must be taken into account for QoS reasons as other services may decide for resource location reasons (i.e. requesting a conversion service located on the same device as the content resource to be transformed would reduce gratuitous bandwidth consumption). On this matter, the plug-in robots must avoid net saturation by producing low priority traffic; performing incremental browsing and subscribing to content update eventing services when supported by the intermediate directory service. Furthermore, the logical representation will avoid replicating container structures present in devices to reduce memory consumption and make the representation homogeneous.

3.3 Content Selection

The user (or a service on behalf of a user) in an Amigo home makes a multimedia request to view certain content on a media renderer. However, content is delivered in a certain format and protocol towards a renderer. This is called the content stream or just stream. Sometimes content itself already contains multiple streams (this is for example possible with MPEG-4 content). An advanced media renderer can then choose the individual stream, based on codec capabilities and/or network bandwidth.

The different streams for a certain content inside an Amigo home are generated by the content adaptation (see 3.4) system. The content selection process matches the content description with the renderer's capabilities to select the optimal stream. The optimal stream is defined as "the stream delivering maximum user perceived quality and consuming minimal bandwidth".

3.3.1 Media Renderer capabilities

The content selection process needs the capabilities of a specific renderer in order to select the optimal stream. An open standard on specifying device capabilities is suggested for the Amigo system like for example CC/PP (Composite Capabilities/Preference Profiles) [CCPP].

However, this standard is currently not widely deployed on multi-media equipment. Legacy multi-media equipment indicates its capabilities:

- Through protocol or vendor-specific extensions (e.g. HTTP or UPnP)
- Not at all.

To allow these legacy devices to still benefit from the content selection process, capability modules are introduced. These capability modules:

- Extract capabilities from protocol specific extensions and translate them to standard CC/PP
- Could be provided by vendors of legacy equipment to transform vendor specific information into standard CC/PP.

- Provide CC/PP information based on a single identifier for a multi-media renderer (e.g. a user agent string in HTTP or a manufacturer/model number in UPnP).

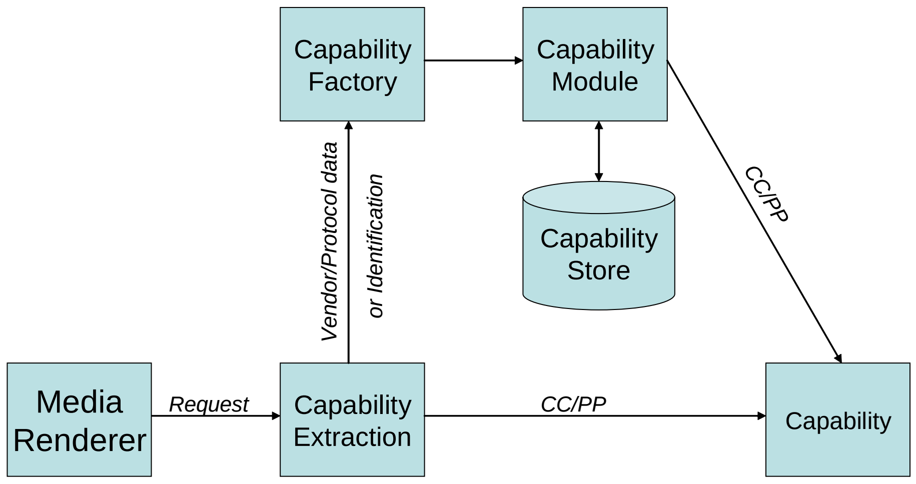


Figure 4: Capability modules

A media renderer’s request is received and its capabilities are extracted. If the request already contained standardized CC/PP capabilities, they are forwarded; otherwise a capability module is instantiated by the capability factory. This capability module will generate the CC/PP information based on protocol extensions or vendor information.

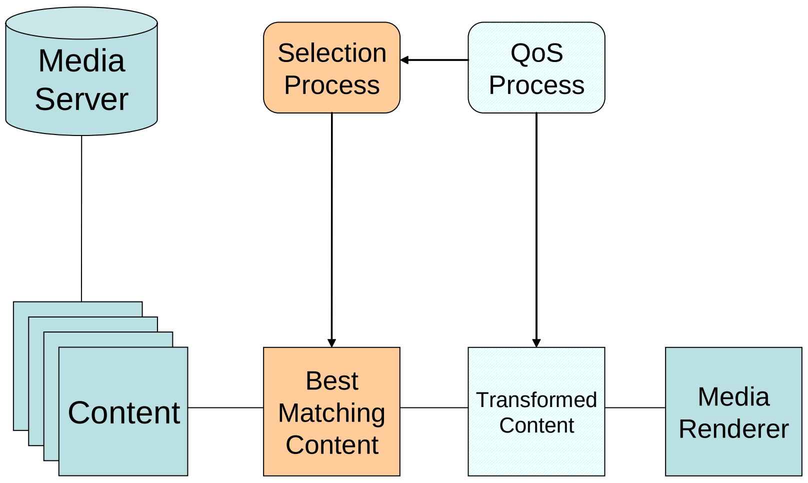
3.3.2 Selection process

Content selection is the matching of renderer capabilities with individual stream description data (see 3.2.1). The goal of this process is to select the content that delivers the maximum perceived quality towards the end-user and requires the minimum network bandwidth. If a device has for example a CIF screen (352 x 288 pixels) and the user would like to watch a TV sized MPEG-2 video on that device, a stream should be selected that is downscaled from the TV format to the CIF format. If that device for example supports MPEG-2 and MPEG-4 streams, an MPEG-4 stream should be selected due to its lesser bandwidth requirements.

If the content selection process is not capable of locating the optimal stream (if in the example above there would be no MPEG-4 stream available of the DVD), it will signal this towards the content adaptation module. The user can then decide to either wait (the content adaptation module might indicate an estimated duration of the transformation) for the transformed content, override the optimal selection or request it at a later point in time.

The content adaptation module will learn the new requested characteristics and prepare new content automatically in this format so that future requests from this device can be automatically selected and delivered.

The decision logic in the content selection module is based on delivering optimal perceived user quality using minimal bandwidth. This logic however could be extended in the future with QoS decisions:



In this schema, the selection process is influenced by the QoS process. Due to QoS policies and available bandwidth, the QoS process might decide to select a different stream and/or transform the stream (e.g. frame-dropping or change of resolution) in order to reduce the required bandwidth.

3.4 Content Adaptation

Inside an AMIGO home there will be multiple devices sharing content, some of them will be AMIGO devices, some of them will be intermediate devices. In both cases, the user must be capable of seamlessly accessing this content in the main display of a room or in his personal device (i.e. PDA, mobile phone…), depending on their preferences, context and availability. Such rendering devices may have radically different capabilities (such as size of the display, installed codecs, whether it is an audio or video device, etc.) and may not accept certain encoding or content formats. A piece of content may be stored as different resources with different formats, even in different media. The middleware will provide means for selecting the appropriate content resources in which a certain piece of content chosen by the user can be rendered in a specified device (see 3.3). Moreover, those resources may not yet exist within Amigo so that presenting the desired content in the desired device will require content resource creation via content adaptation. Furthermore, certain content items may not even be in the media a certain device is capable of rendering; that could be the case for content that is text and which the user may request AMIGO to read out aloud.

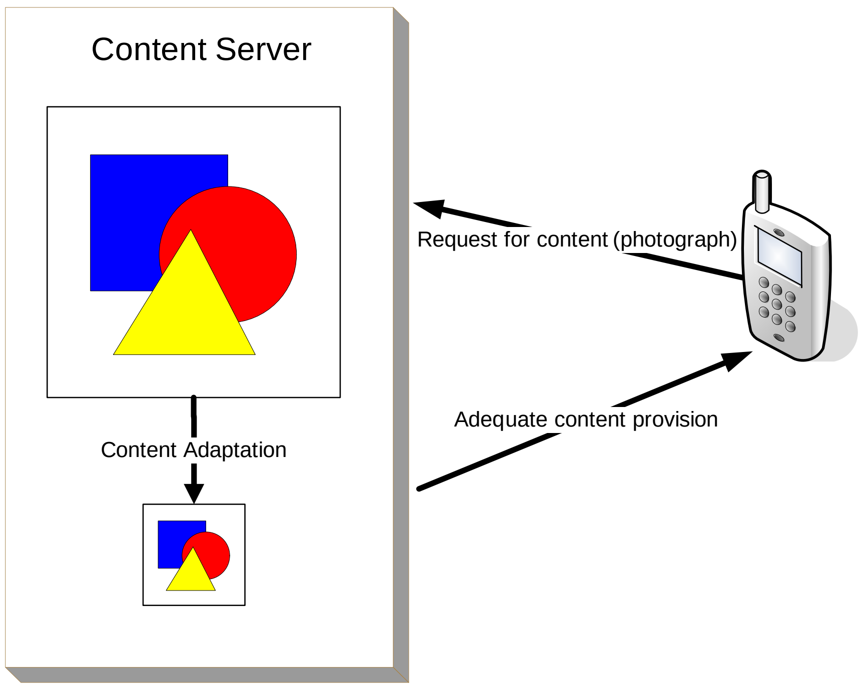


Figure 5: User accesses an image oversized for the terminal display and the server performs content adaptation

Figure 5 describes a typical scenario where a user wants to access a photograph, located at an Amigo Content Server via his mobile phone. The mobile phone has a limited screen resolution so the stored image cannot be viewed. The middleware should then detect the format discordance between the resource and the device capabilities and convert the content (if the needed plug-in is available), to the maximum resolution accepted by the mobile phone to optimize visualization quality and user experience.

DLNA guidelines specify that a Digital Media Renderer (DMR) claiming support for a certain media class must support the DLNA mandatory formats for that media class (see chapter 6 in [D2.1]). However, this cannot be afforded by many devices outside and inside the CE domain that may have limited processing or displaying capabilities (i.e. mobile phones or other legacy devices). Therefore a content adaptation mechanism that can support conversion to different formats other than those mentioned is needed, even more if QoS issues could indicate one of these formats as (in/more) appropriate.

3.4.1 Architecture Overview

A content object represents a portion of multimedia content that may be available as a variety of resources. The resources, although representing the same content, may have different formats (i.e. a video fragment in different resolutions or encodings). The content object may contain metadata providing further semantic and syntactic information. This metadata can provide a description of the resources available for a certain object, so that a selection mechanism as the one proposed above may proceed.

Content resources use storage space and are therefore limited within a server. Thus, it is impossible to have an appropriate resource for every possible set of characteristics a rendering device could have. Due to the variety of devices in the Amigo home, the selection mechanism will certainly sometimes fail to find an adequate resource for a certain device. In this case, we can confirm that *the resource should be transformed into a compatible format and delivered if the conversion delay is acceptable.*

To achieve this goal, a set of appropriate conversion tools must be present at the server or published as Amigo services, and the Amigo middleware must be able to select at least one that can perform the desired transformation. It is probable that a device accessing content in the Amigo home belongs to the devices usually connected to the home network from the inside or the outside of the home, especially if the device is not registered as a guest device (Chapter 8 in [D2.1]). Therefore, if the time required for the conversion is not accepted by the user (explicitly or implicitly based on preferences), having the same content adapted to this device for subsequent access would minimize the exasperating user experience of having to wait for content access. An extreme example is the case of a film stored in a different format than that supported by the main living room display. Video conversion is a heavily processor consuming operation that can increase waiting times to hours: a solution to this situation is transcoding, but it represents an enormous processing load and a real time environment, which would require some kind of dedicated system.

The proposed solution to this problem is a formatted resource cache. This cache will store every converted content resource for a variable time assigned according to a *perishing policy*. The converted content will henceforth be available for future accesses by devices accepting this new format as the optimal one, reducing adaptation delays and processing load. Thus, the cache will make its resources available for the content selection process queries. The content adaptation architecture is depicted in Figure 6.

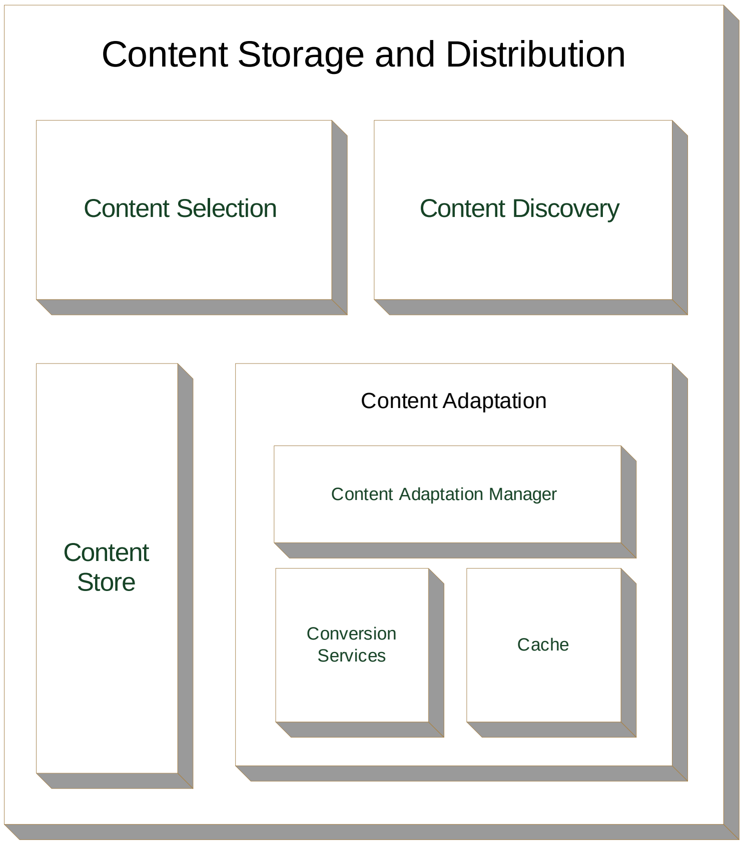


Figure 6: content adaptation architecture within Content Storage and Distribution

3.4.2 Content Adaptation Manager

The content adaptation module will provide a set of services to the rest of the content storage and distribution modules. The possibility of finding an existing conversion service in the home network is of interest for devices that may require conversion of the content they hold but cannot afford to host due to their limited processing capabilities. Therefore, the availability enable other applications or services outside the content storage and distribution module to access some of the content adaptation services must be accomplished. Furthermore, if conversion times of certain types of content such as video are taken into account, implementing services that would notify other services and applications of adaptation events becomes a requirement, as it would significantly minimize traffic and processing load produced by polling techniques. Therefore, the functionalities of this module are publication of those services outside the content adaptation module and related event notification to registered entities, as well as redirecting to the cache invocation to conversion services that would result in retrieving an existing cached resource.

Three main services can be directly identified at different levels; they will be further explained in the sections below:

- *Conversion Services* furnished with event notification (e.g. conversion completion notification).
- *Cache Browsing Services* so that converted instances of the same content may be discovered by applications and services, since a cached converted instance of content may exist in a different device to that hosting the original content.
- *Conversion Plug-in Registration Services* so that conversion plug-ins can be registered by other entities.

Eventually, when matching device capabilities to content format, Content Selection (see 3.3) will require content adaptation. This is done through the *Content Adaptation Manager* which will check first for existing instances of the requested resource in the cache and if not available redirects the request to the conversion services. The content selection module may require notification on conversion completion that is provided by the *Content Adaptation Manager* interface.

3.4.3 Adapted Content Cache

The architecture overview of the cache is presented in Figure 7.

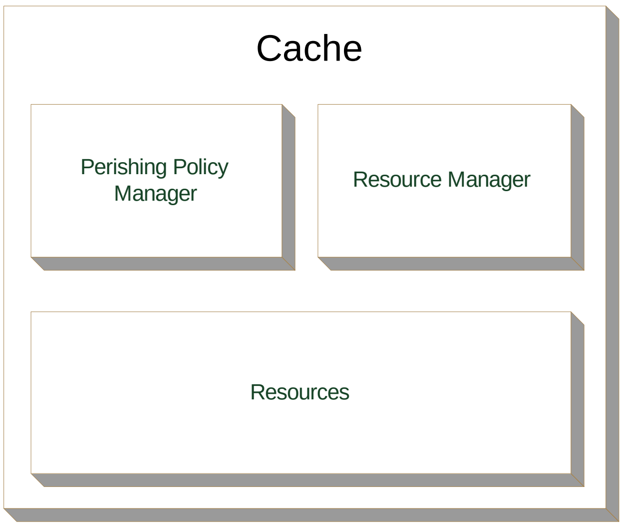


Figure 7: Cache architecture overview

In Figure 7, the *Resource Manager* is responsible for resource registration in the cache, resource retrieval and resource access as well as storage issues (i.e. storage space in the cache). The *Perishing Policy Manager* is responsible for stating persistency levels and associating them to some policy, the latter including expiration time, expiration delay due to access and persistency level promotion. Both modules must work closely together, the *Resource Manager* informing the *Perishing Policy Manager* of content registration and receiving notifications from it whenever a resource *expiration time* arrives.

Therefore, the *Perishing Policy Manager* interface to the *Resource Manager* will specify:

- A policy request service which will associate a policy with a resource.
- An eventing service which will conduct notification of policy related events to the *Resource Manager*.
- A policy update service which will enable changes of policy or simply provide information to the *Perishing Policy Manager* about resource use, importance, etc.

The *Resource Manager* will provide resource access information to the *Perishing Policy Manager* to update policies.

Example: let us suppose that a newly converted resource is stored in the cache. This will be done through the *Resource Manager* storage services. The *Resource Manager* will perform some functions such as checking for available storage space and, after success, will call the *Perishing Policy Manager* policy request service to assign a policy to that resource inside the *Perishing Policy Manager* and establish an eventing link for that resource. Then, the resource will be registered inside the *Resource Manager* and finally stored. As the converted resource is accessed through the *Resource Manager* retrieval service, the *Perishing Policy Manager* will receive reports from the *Resource Manager* that will update associated perishing dates or the whole associated policy. If a perishing event occurs (e.g. the perishing date has arrived) then the *Perishing Policy Manager* will inform (through the eventing link), the *Resource Manager* which will take action (e.g. delete the resource).

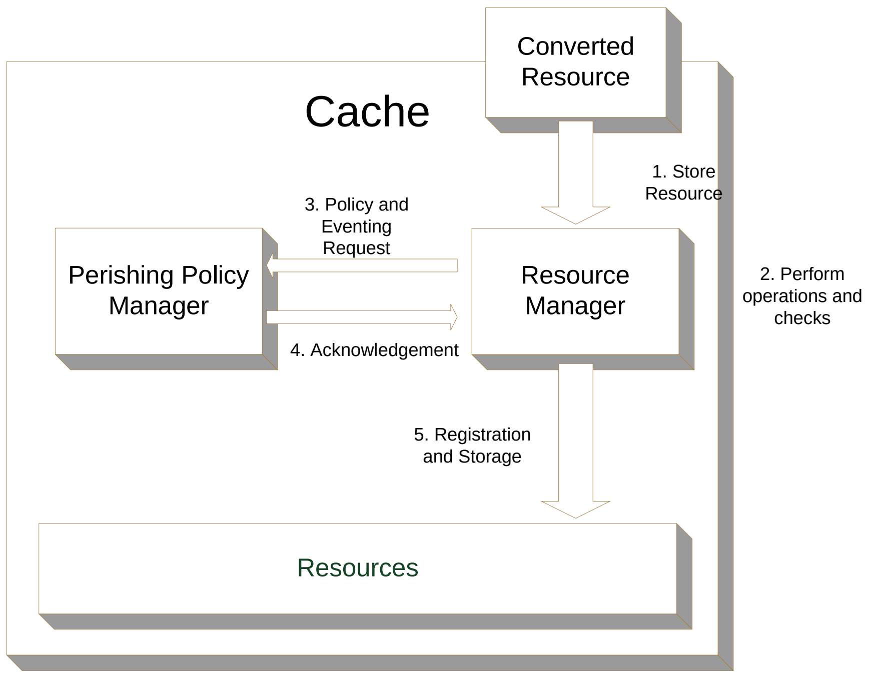


Figure 8: Resource registration in the cache

3.4.4 Conversion Services

3.4.4.1 Content Conversion

In order to find the conversion tool that converts from one of the available formats to the required format, the conversion tool must be associated with a description or a set of descriptions. Formally, a generic conversion tool is a function $F: C \times P \rightarrow C$ where C is the set of all possible content resources, so that if r is a content resource then $r \in C$, and P is a set of parameters that will be refined below. The first set can be divided in different subsets representing the content media (A for audio, V for video, I for images, T for text,…):

$$C \equiv A \cup V \cup I \cup T \cup \ldots$$

In turn each subset can be defined as the union of subsets grouping resources in the same encoding, for example:

$$A = \bigcup_i A_i$$

Where A_i represent an audio encoding (i.e. A_1= "Audio PCM", A_2 = "Audio MP3"). A particular audio encoding depends on a set of parameters, some of which will be common to all content resources (i.e. resource size), to some (i.e. temporal sampling rate), to that of the same media subset (i.e. number of channels) and some specific to that encoding (i.e. JPEG quality). Let P be the set of all these parameter sets and let P_{Ci} be the subset of these that have sense within a subset of content resources $C_i \subset C$. Then, a specific conversion function can be defined as:

$$f\colon C_i \times P_{Ci} \times P_{Co} \rightarrow C_o \text{ so that } f(r, p_i, p_o) = y\,;\ C_i \subset C,\ C_o \subset C,\ p_i \subset P_{Ci},\ p_o \subset P_{Co}\,.$$

Thus, a conversion tool can be modeled as a set of these functions, each of which provide a transformation from a certain content subset to the same or another subset receiving the input and output encoding parameters for the input and output subsets. It is typical that these content subsets (C_i , C_o) coincide with encoding subsets (i.e. *A*, *V*, *I*,…). Moreover, a certain transformation function may support only a subset of the input and output parameter values.

It would be necessary then to investigate on the parameter sets associated to the encoding subsets and the permitted values, so that a format description language can be defined. This language will provide the basis for a transformation selection mechanism as it would describe source and destination formats, and with them, the essence of the transformation itself.

3.4.4.2 Multimedia Format Conversion Factory

There is a wide spectrum of software for multimedia format conversion, some more specialized some more general, ranging from open source and standardized to closed source proprietary products. Providing a set of format conversion tools lays out the scope of the topic, although format conversion is required to achieve interoperability at certain level. Here, a hybrid solution is presented which relies on plug-ins for actual conversion, but establishes a framework for the multimedia conversion software developer or integrator, and provides the necessary mechanisms to easily integrate conversion plug-ins in Amigo.

The proposed format conversion services architecture is depicted in Figure 9. A *Conversion Factory* will manage conversion requests redirecting them to the corresponding plug-in. This will be attained by querying the *Plug-in Registry* which will contain the registered conversion plug-in descriptions in the *Conversion Function Description Registry*. The *Plug-in Registry* will offer *Plug-in Registration Services* to allow plug-in registration in the middleware.

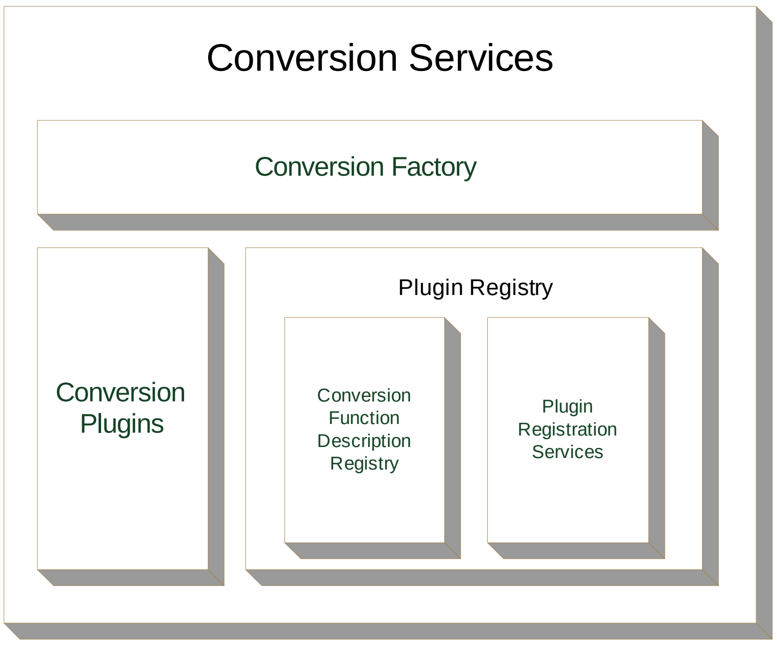


Figure 9: Conversion services architecture

Due to the constant appearance of new formats and codecs in the multimedia domain, the proposed architecture intends to be as independent as possible from a closed set of currently existing formats. In order to do so, the *Conversion Factory* and the *Plug-in Registry* will manage conversion function queries (see previous section) transparently. The language in which the queries will be expressed set the limits of the adaptability of the whole architecture if the conversion function descriptions and the conversion function queries are expressed in the same language. Therefore an extensible language for expressing content format and, therefore, conversion functions (i.e. a pair of formats: input and output formats) will imply an extensible architecture. The basis for this language is addressed in the content description section below.

The Plug-in interface, which will be defined more specifically, will have the resource and a transformation description as input parameters, which will be understandable by the plug-in if the registry information is coherent (as the latter would contain at least this transformation description, otherwise this plug-in would not have been selected). The *Plug-in Registration Services* interface will have at least the plug-in description as an input. The transparent management of the conversion function descriptions and queries permits conversion plug-in renewal and modification with no architecture or code modification. It would also support appearance of new formats and conversion tools between them just by extending the language.

The definition of the Plug-in interface and the language for Conversion function descriptions would provide the developer and integrator the necessary framework for developing or integrating multimedia format conversion software in Amigo.

3.5 Conclusion

The proposed architecture enables content rendering on the widest range of media renderers through a content selection and adaptation framework. Plug-ins can be provided for standardized protocols and format and Vendors of media renderers and content distributors can provide plug-ins that are used to supply content in that specific format to optimally suit the devices in the home. Different kinds of plug-in are needed (content format, renderer capability, content adaptation) that can all be packaged and Amigo standardized. In this way, vendors and content owners can provide this Amigo Content Plug-in package to make sure their devices and content can be optimally experienced inside an Amigo home.

4 Data Storage

4.1 Introduction

The need for a middleware component handling reliable cross-application data retrieval and storage was identified during a WP1 to WP2 result-handover workshop. It is clear from the analysis in [D1.2] that the system should provide means for secure and reliable storage of data instead of implicitly relying on the storage capabilities and qualities imposed by an application and its hosting device. The scenario of a user using his PDA to store all his personal information and losing it after a crash or battery problem is maybe the most representative one although there are other situations that call for a system wide storage solution: equipment (holding data) is replaced or sold, inhabitants move to a different location. Besides these user requirements there are technical requirements that justify a system-wide storage solution: A lot of information inside an Amigo system is exchanged between different components (in the middleware and application layers). Flexibility, Extensibility and Reliability are qualities that benefit from a system-wide storage solution. Resource-constrained devices can use the storage facility to increase reliability and reduce system resource consumption. Applications can use it to share common or important data whilst new applications can easily use the existing data structures.

4.2 Requirements

The requirements are composed from the specific storage (user) requirements in [D2.1] and the (technical) requirements from the refined Amigo scenarios in [D2.1].

Interoperable
Access to the data store should be possible from heterogeneous services and platforms.

Generic
The solution should not impose restrictions and limitations on the structure of the data that can be stored.

Dynamic
Services should be able to create, modify and delete new stores inside the solution at runtime without interrupting the storage functions of existing stores.

Versioning
A store should be versionable and enable retrieval of version snapshots or their data (e.g.: retrieve the data as it was on 1-1-2005).

Notification
It should be possible to subscribe to notifications (data added, data deleted, data modified) for a specific store inside the data storage solution.

Secure
The data of a store should be secured in such a way that the owner of the data can decide who has what kind of access to the data in a store.

Availability & Reliability
The access to the data inside the store must be continuously available (except during restore/recovery operations). The storage solution should incorporate mechanisms for automatic recovery from hardware and/or power failures.

4.3 Design

The different roles and entities that relate to the data store solution are shown in the figure below. It should be noted, however, that the roles of User and Creator are usually played by components (middleware or application layer) in the Amigo system, not physical users.

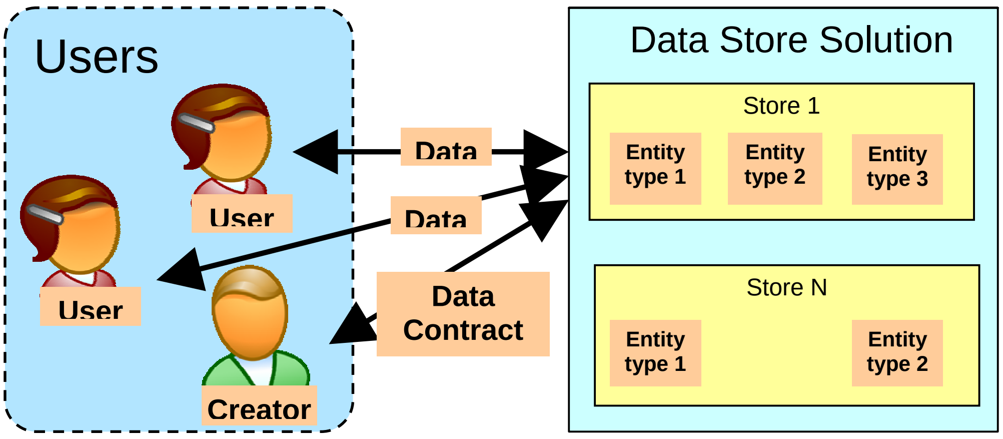


Figure 10: Roles and Entities in the data storage solution

The Creator is the one who initially creates a store in the data store solution by providing a contract. The contract contains a description of the data structure (each item in the store is called an entity), the ACL and the notifications that Users can subscribe to. Users can receive this contract to inspect the data structure and/or notifications that they can subscribe to. The data contract may also contain additional information related to the versioning and maintenance of the store.

Users can, besides retrieving the store's contract, access (add, delete or modify) the data in the store according to the ACL. Using the information in the contract and the ACL, users can subscribe to notification from the data store solution.

The data store solution is a centralized component hosting the different stores. Each type that can be stored is called an entity (the structure of each entity is defined in the data contract). An example of an entity is "Persons" containing some fields like "Name", "Gender", "Date of birth" etc. Each individual record is called an entity instance (for example "Maria"). Note that the same store may also contain additional entities like for example "Groups". The data store solution also takes care of the maintenance and restore/recovery procedures for the individual stores.

4.3.1 Interoperability

Interoperability for the data store solution is guaranteed by using technology and platform independent standardized web services. The data contract will be specified using [XML] language. Notification will be done using [WS-Eventing]. The data store solution will be implemented as a secured service that secures communication (with secured services) and identification but does not provide access control. This allows also unsecured or legacy services to use the data store solution. Resource constrained devices that do not have the capability to use web service technologies might still use the data store through a custom proxy service that uses a proprietary protocol towards the devices and the web service protocol towards the data store (note that this proxy will not be part of the data store).

4.3.2 Generic

The data structure of an individual store is specified as part of the contract in [XSD]. This allows the full usage of all available simple types and the custom construction of complex types.

4.3.3 Dynamic

The data store solution allows creation, deletion and modification of individual stores at runtime. This does not affect other stores inside the data store solution.

4.3.4 Versioning

Whether versioning inside a store is used is specified in the Data Contract by the Creator. If versioning is activated, every time an entity is changed (added, deleted or modified), a snapshot of that entity is stored (up to a maximum configurable version depth). A timestamp is used to retrieve historical (versioned) data.

4.3.5 Notification

The notifications that are supported by a store are specified in the data contract by the creator. There are 2 different levels of notification: on entity level (meaning that notifications are received for modification on **any** entity instance; e.g. “Persons”) and on entity instance level (meaning that a notification is received if a specific instance; e.g. “Maria”).

Since data contracts are managed by an internal store (that is external readable), notifications to data contracts (either all or a specific data contract) are also supported.

4.3.6 Secure

The data store solution itself is a so called “secured service” (see Security & Privacy in [D2.1]). However, the security architecture will not be used for access control; hence the data store solution will not show up in the Authorization Scheme [D2.1]. The security tokens can still be used for (1) communication security to encrypt privacy sensitive data and (2) verifying the identity of the data store solution or the user. In this way, also unsecured services can use the security service.

When a store is created with an ACL, the identity of the creator is associated with that store. The ACL (specified by the creator in the data contract) is used to control who and how can access that individual store.

4.3.7 Availability & Reliability

The data store solution is implemented using a replication mechanism to increase the availability of the solution. This replication mechanism is hidden from the user.

The data store solution provides automatic functions for backup and restore of individual stores.

4.4 Conclusion

The data store solution is a secure, generic, system-wide solution for sharing and reliable storing data. If offers its services to secured as well as unsecured services and uses standard platform-independent protocols to communicate with users. Secured services have the additional functionality of using an ACL to control who (besides themselves) can use and in what way they can use their data. Secured services can use secured communication to transmit their (privacy-sensitive) data. It also reduces network traffic and device resources by providing notifications on data or data contract changes. Versioning can be used to retrieve historical data (“retrieve address book from last year”, or “retrieve context data from 1 hour ago”).

5 Conclusion

The goal of the Amigo project is to merge automation in the traditional separated domains of mobile, personal computing, home automation and Consumer electronics into the networked home. This implies heterogeneous devices, platforms and technologies. If we want the end-user to embrace this new networked home, he first has to trust that it will do what it promises to do; that the system protects his privacy and that although many functions are now controllable from many sources, the end-user is still and always will be in control.

The security & privacy design elaborated in this document and [D2.1] was specifically designed for this purpose. It provides an easy and natural way towards the end-user for configuration without sacrificing security and/or functionality. The components described in chapter 2 enable application developers to build secure applications that easily fit into a new or existing Amigo system.

Data can be protected against unauthorized access by using the Data store component described in chapter 4. In addition, this component also increases the reliability of the system by providing automatic means for data recovery in case of crashes or other situations that could cause data loss.

The content distribution design enables seamless playback of any content on any device by adapting the content to the device capabilities. Its self-learning capabilities reduce waiting times as well as bandwidth consumption to a minimum.

Acronyms

AAS	Authentication Authorization Service
ACL	Access Control List
AV	Audio Video
CBC	Cipher Block Chaining
CC/PP	Composite Capabilities/Preference Profile
CDDB	Compact Disc Database
CIF	Common Intermediate Format
DIDL	Digital Item Declaration Language
DMR	Digital Media Renderer
DLNA	Digital Living Network Alliance
DVD	Digital Video Disc
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
JENA	a Java API for RDF
Jini	a service discovery protocol for Java
MAC	Media Access Control
MP3	Moving Picture Experts Group Layer-3 Audio
MPEG	Motion Picture Experts Group
OWL	Ontology Web Language
QoS	Quality of Service
RBAC	Role Based Access Control
RDF	Resource Description Framework
RFID	Radio Frequency Identification
RMI	Remote Method Invocation
SOAP	Simple Object Access Protocol
SSDP	Simple Service Discovery Protocol
TGT	Ticket Granting Ticket
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
WAV	Windows Wave

References

[CCPP] Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0, http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/, January 2004

[D1.2] Amigo project deliverable D1.2, Report on User Requirements, Volume I-III, February 2005

[D2.1] Amigo project deliverable D2.1, Specification of the Amigo Abstract Middleware Architecture, February 2005

[D3.1a] Amigo project deliverable D3.11a, Detailed design of base Amigo core middleware infrastructure Detailed, part a, September 2005

[D3.1b] Amigo project deliverable D3.1b, Detailed design of base Amigo core middleware infrastructure Detailed, part b, September 2005

[JMM04] José M. Martínez. MPEG-7 Overview (version 10). Palma de Mallorca, October 2004. http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm

[KoNe92] J. Kohl and C. Neumann, The Kerberos Network Authentication Service (V5), RFC1510, Internet Engineering Task Force, 1993.

[RBAC] D.F. Ferraiolo and D.R. Kuhn, "Role Based Access Control", 15th National Computer Security Conference, 1992.

[TripleDES] Data Encryption Standard (DES), Federal information processing standards publication 46-3, October 1999

[UPnPDS03] Device Security V1.0, UPnP Forum, November 2003, http://www.upnp.org/standardizeddcps/documents/DeviceSecurity_1.0cc_001.pdf

[WSD05] Web Services Dynamic Discovery (WS-Discovery), April 2005, http://msdn.microsoft.com/ws/2005/04/ws-discovery/

[WS-Eventing] Web Services Eventing (WS-Eventing), August 2004, http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnglobspec/html/ws-eventing.asp

[WSS04] Web Services Security: SOAP Message Security 1.0 (WS-Security 2004), OASIS Standard 200401, March 2004, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[XML] Extensible Markup Language (XML) 1.0 (Third Edition), February 2004, http://www.w3.org/TR/2004/REC-xml-20040204/

[XSD] XML Schema Part 0: Primer Second Edition, October 2004, http://www.w3.org/TR/xmlschema-0/