Network Working Group
Request for Comments: 2244
Category: Standards Track

C. Newman
Innosoft
J. G. Myers
Netscape
November 1997

ACAP -- Application Configuration Access Protocol

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

The Application Configuration Access Protocol (ACAP) is designed to support remote storage and access of program option, configuration and preference information. The data store model is designed to allow a client relatively simple access to interesting data, to allow new information to be easily added without server re-configuration, and to promote the use of both standardized data and custom or proprietary data. Key features include "inheritance" which can be used to manage default values for configuration settings and access control lists which allow interesting personal information to be shared and group information to be restricted.

Table of Contents

ACAP Protocol Specification

1. Introduction

1.1. Conventions Used in this Document

In examples, "C:" and "S:" indicate lines sent by the client and server respectively. If such lines are wrapped without a new "C:" or "S:" label, then the wrapping is for editorial clarity and is not part of the command.

The key words "REQUIRED", "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [KEYWORDS].

1.2. ACAP Data Model

An ACAP server exports a hierarchical tree of entries. Each level of the tree is called a dataset, and each dataset is made up of a list of entries. Each entry has a unique name and may contain any number of named attributes. Each attribute within an entry may be single valued or multi-valued and may have associated metadata to assist access and interpretation of the value.

The rules with which a client interprets the data within a portion of ACAP's tree of entries are called a dataset class.

1.3. ACAP Design Goals

ACAP's primary purpose is to allow users access to their configuration data from multiple network-connected computers. Users can then sit down in front of any network-connected computer, run any ACAP-enabled application and have access to their own configuration data. Because it is hoped that many applications will become ACAP-enabled, client simplicity was preferred to server or protocol simplicity whenever reasonable.

ACAP is designed to be easily manageable. For this reason, it includes "inheritance" which allows one dataset to inherit default attributes from another dataset. In addition, access control lists are included to permit delegation of management and quotas are included to control storage. Finally, an ACAP server which is conformant to this base specification should be able to support most dataset classes defined in the future without requiring a server reconfiguration or upgrade.

ACAP is designed to operate well with a client that only has intermittent access to an ACAP server. For this reason, each entry has a server maintained modification time so that the client may detect changes. In addition, the client may ask the server for a list of entries which have been removed since it last accessed the server.

ACAP presumes that a dataset may be potentially large and/or the client's network connection may be slow, and thus offers server sorting, selective fetching and change notification for entries within a dataset.

As required for most Internet protocols, security, scalability and internationalization were important design goals.

Given these design goals, an attempt was made to keep ACAP as simple as possible. It is a traditional Internet text based protocol which massively simplifies protocol debugging. It was designed based on the successful IMAP [IMAP4] protocol framework, with a few refinements.

1.4. Validation

By default, any value may be stored in any attribute for which the user has appropriate permission and quota. This rule is necessary to allow the addition of new simple dataset classes without reconfiguring or upgrading the server.

```
   url-attr-list      = url-enc-attr *("&" url-enc-attr)

   url-auth           = ";AUTH=" ("*" / url-enc-auth)

   url-achar          = uchar / "&" / "=" / "~"
                          ;; See RFC 1738 for definition of "uchar"

   url-char           = uchar / "=" / "~" / ":" / "@" / "/"
                          ;; See RFC 1738 for definition of "uchar"

   url-enc-attr       = 1*url-char
                          ;; encoded version of attribute name

   url-enc-auth       = 1*url-achar
                          ;; encoded version of auth-type-name above

   url-enc-entry      = 1*url-char
                          ;; encoded version of entry-relative above

   url-enc-user       = *url-achar
                          ;; encoded version of login userid

   url-extension      = *("?" 1*url-char)

   url-filter         = "?" url-attr-list

   url-relative       = url-acap / [url-enc-entry] [url-filter]
                          ;; url-enc-entry is relative to base URL

   url-server         = [url-enc-user [url-auth] "@"] hostport
                          ;; See RFC 1738 for definition of "hostport"
```

9.   Multi-lingual Considerations

The IAB charset workshop [IAB-CHARSET] came to a number of conclusions which influenced the design of ACAP. The decision to use UTF-8 as the character encoding scheme was based on that work. The LANG command to negotiate a language for error messages is also included.

Section 3.4.5 of the IAB charset workshop report states that there should be a way to identify the natural language for human readable strings. Several promising proposals have been made for use within ACAP, but no clear consensus on a single method is apparent at this stage. The following rules are likely to permit the addition of multi-lingual support in the future:

(1) A work in progress called Multi-Lingual String Format (MLSF) proposes a layer on top of UTF-8 which uses otherwise illegal UTF-8 sequences to store language tags. In order to permit its addition to a future version of this standard, client-side UTF-8 interpreters MUST be able to silently ignore illegal multi-byte UTF-8 characters, and treat illegal single-byte UTF-8 characters as end of string markers. Servers, for the time being, MUST be able to silently accept illegal UTF-8 characters, except in attribute names and entry names. Clients MUST NOT send illegal UTF-8 characters to the server unless a future standard changes this rule.

(2) There is a proposal to add language tags to Unicode. To support this, servers MUST be able to store UTF-8 characters of up to 20 bits of data.

(3) The metadata item "language" is reserved for future use.

10.   Security Considerations

The AUTHENTICATE command uses SASL [SASL] to provide basic authentication, authorization, integrity and privacy services. This is described in section 6.3.1.

When the CRAM-MD5 mechanism is used, the security considerations for the CRAM-MD5 SASL mechanism [CRAM-MD5] apply. The CRAM-MD5 mechanism is also susceptible to passive dictionary attacks. This means that if an authentication session is recorded by a passive observer, that observer can try common passwords through the CRAM-MD5 mechanism and see if the results match. This attack is reduced by using hard to guess passwords. Sites are encouraged to educate users and have the password change service test candidate passwords against a dictionary. ACAP implementations of CRAM-MD5 SHOULD permit passwords of at least 64 characters in length.

ACAP protocol transactions are susceptible to passive observers or man in the middle attacks which alter the data, unless the optional encryption and integrity services of the AUTHENTICATE command are enabled, or an external security mechanism is used for protection. It may be useful to allow configuration of both clients and servers to refuse to transfer sensitive information in the absence of strong encryption.

ACAP access control lists provide fine grained authorization for access to attributes. A number of related security issues are described in section 3.5.

ACAP URLs have the same security considerations as IMAP URLs [IMAP-URL].

ACAP clients are encouraged to consider the security problems involved with a lab computer situation. Specifically, a client cache of ACAP configuration information MUST NOT allow access by an unauthorized user. One way to assure this is for an ACAP client to be able to completely flush any non-public cached configuration data when a user leaves.

As laptop computers can be easily stolen and a cache of configuration data may contain sensitive information, a disconnected mode ACAP client may wish to encrypt and password protect cached configuration information.

11. Acknowledgments

Many thanks to the follow people who have contributed to ACAP over the past four years: Wallace Colyer, Mark Crispin, Jack DeWinter, Rob Earhart, Ned Freed, Randy Gellens, Terry Gray, J. S. Greenfield, Steve Dorner, Steve Hole, Steve Hubert, Dave Roberts, Bart Schaefer, Matt Wall and other participants of the IETF ACAP working group.

12. Authors' Addresses

Chris Newman
Innosoft International, Inc.
1050 Lakes Drive
West Covina, CA 91790 USA

Email: chris.newman@innosoft.com

John Gardiner Myers
Netscape Communications
501 East Middlefield Road
Mail Stop MV-029
Mountain View, CA 94043

Email: jgmyers@netscape.com