HL7 Context Management "CCOW" Standard: Best Practices and Common Mistakes, Version 1.0, May 2006

Editor: CCOW Technical Committee

Health Level Seven and HL7 are trademarks of Health Level Seven, Inc.

Table of Contents

1 Introduction

This document provides a collection of non-normative recommended best practices, descriptions of common mistakes, and implementation reminders for application developers to consider when creating CCOW-compliant applications. The content of this document pertains to CCOW Version 1.5, but in general is applicable to all preceding versions of CCOW as well.

1.1 CCOW OVERVIEW

The Health Level Seven Context Management Standard (CMS) defines a means for the automatic coordination and synchronization of disparate healthcare applications that co-reside on the same clinical desktop. Applications that use the CMS standard enable the user to set the clinical context for the desktop using any of the enabled applications. When the context has been set, all of the enabled applications on the desktop are automatically "tuned" to the same clinical context.

The clinical context is comprised of a set of clinical context subjects. Each subject represents a real-world entity, such as a particular patient, or concept, such as a specific encounter with a patient. The CMS standard defines several standard subjects and allows non-standard (or custom) subjects to be defined as well.

By sharing context, applications are able to work together to follow the user's thoughts and actions as they interact with a set of applications. These applications are said to be "clinically linked." Working together in concert, this cooperative behavior among the applications makes it much easier and safer for users to enter and retrieve the information that they need to deliver care to their patients.

The CMS is extremely prescriptive, but as it is only a standard it can only go so far in terms of guiding how applications are actually designed and implemented. Variability among the decisions that application developers make can lead to various amounts of confusion for users of multiple independently-developed CCOW-compliant applications. In order to address this situation this document offers a series of recommendations that, when followed by application developers will produce a cohesive and uniform set of CCOW-compliant behaviors.

1.2 HOW TO USE THIS DOCUMENT

The foremost goal of this document is to provide recommendation that will enable CCOW-compliant applications do implement basic CCOW capabilities in the same way. Applications can do more sophisticated things than are recommended in this document, but they should do so only in addition to, as opposed to instead of, doing the basic things as specified herein.

In other words, application developers are encouraged to do the right things to make their applications behave consistently with other applications per this document. If an alternative behavior is provided then it is recommended that application users be provided with a means to disable the alternative behavior via a configuration switch so that only the recommended behavior is achieved.

Note that when a section in the CMS is cited, it is version 1.5 that is being referred to. The following document name abbreviations are used:

CMA = Health Level-Seven Standard Context Management Specification, Technology and Subject-Independent Component Architecture, Version CM-1.5

SDD = Health Level-Seven Standard Context Management Specification, Subject Data Definitions, Version CM-1.5

UIS = Health Level-Seven Standard Context Management Specification, User Interface: Microsoft Windows and Web Browsers, Version CM-1.5

ATM = Health Level-Seven Standard Context Management Specification, Component Technology Mapping: ActiveX, Version CM-1.5

WTM = Health Level-Seven Standard Context Management Specification, Component Technology Mapping: Web, Version CM-1.5

1.3 CCOW-RELATED APPLICATION CONFIGURATIONS

Throughout this document the concept of CCOW-related application configurations are discussed. All of these configurations are intended to be set centrally for the application by an appropriately authorized systems administrator, as opposed to being capabilities that can be controlled by an end user.

2 Best Practices

2.1 MEANS OF LAUNCHING APPLICATIONS

Issue: Some applications require a different means for being launched as the technique for determining whether or not the application will attempt to join a context session. (For example, an application might be launched using different URLs, one which instructs the application to join a context session, the other which instructs the application to run independently.) This creates unnecessary administrative and configuration complications for both IT administrators and end-users in enterprises where some workstations provide CCOW-compliant capabilities and other workstations do not, as applications that implement this behavior need to be set-up differently and launched differently on the different types of workstations.

Recommendation: Applications should not require different ways to be launched as the means to elicit CCOW or non-CCOW behavior. Instead applications should adhere to recommendation 2.2.

2.2 APPLICATION START-UP

Issue: When launched, some applications do not check to see if the context manager is available. These applications then do not function properly if in fact the context manager is not available.

Recommendation: Applications should be configurable as to whether or not they should run in a CCOW context. If an application is configured to not run in the context, then no CCOW icons should be displayed and no attempt to reference the CCOW infrastructure should be made. However, if configured to run in the CCOW context, then the CCOW link icon should always be displayed.

The remainder of this recommendation pertains only to those applications configured to run in the CCOW context.

Upon being launched, applications should check for the context manager and join the context if the context manager is available. The techniques for doing this depend upon the technology with which the application is implemented, but are well-defined in ATM and WTM. If the application successfully joins the context (i.e. the necessary CCOW infrastructure is available), then the application should proceed as normal.

If the application does not successfully join the context (i.e. the CCOW infrastructure is not available), then the application should continue to run in a stand-alone manner if possible and the CCOW icon should be shown in the link-broken state. The user should be informed about the situation by the application via a site-configurable text that allows at least 1024 characters, followed by an application-provided text that describes the action required to rejoin the context. The recommended action for the application to rejoin the context (or at least to attempt to rejoin) is to have the user take an explicit action such as selecting a “join context” menu item or toggling the CCOW icon.

2.3 SUPPORT FOR CITRIX / WTS

Issue: Applications should be constructed so they can be accessible via Citrix and Windows Terminal Server.

Recommendation: In the method call ContextManagementRegistry::Locate, applications should always set descriptive data per CCOW Component Technology Mapping Specification for Web/HTTP in order to enable support for Citrix and WTS. Specifically, applications should set the ContextManagementRegistry::Locate input parameter descriptiveData using the citrixSessionId format if the application is running in a Citrix-hosted environment or wtsSessionId if the application is running in a WTS-hosted environment. The information embedded in the citrixSessionId and wtsSessionId formats is

more reliable than the clientHostName or clientIPAddress formats. Note that this recommendation pertains only to applications implemented via the WTM.

2.4 REJOINING THE CONTEXT

Issue: When re-joining a context session after being suspended, some applications set the context rather than tuning to the context as it presently stands. This causes surprising and unexpected behaviors for users.

Recommendation: In order to offer consistent and uniform behavior, applications should automatically tune to the current context when re-joining the context or resuming. Applications should not set the context when re-joining or resuming. Note that the UIS currently states that the user should be given the option to set the context in either direction. The recommendation here is put forth in order to reduce the complexity of the user interface with regard to context navigation and management.

2.5 CHANGING THE CONTEXT VIA A SCROLLABLE LIST

Issue: Applications frequently have lists of patients, observations, encounters, or other objects from which users make selections for further viewing or other actions. Some applications attempt to set the context even when the user is actively scrolling through the list but do not actually tune to the context changes itself. The result is that context is set many times very rapidly by the applications behavior in response to the user's scrolling action. This can create a substantial and unnecessary context change transaction performance load on a system of applications that share a common context.

Recommendation: If an application offers a scrollable list of context sensitive objects, and the user does not expect to change the context until having successfully scrolled to the intended item, then the following practice is recommended. The context should only be set by the application after X seconds have transpired since the user stopped scrolling through the list, where X is configurable on an application-specific basis. In addition to, or instead of this timer, the application may also set the context when the user has clearly indicated a selection by clicking on a mouse or performing an equally explicit gesture. Note that an application that implements this behavior must be prepared to cancel the selection if user instructs that the context change be cancelled, and the application must revert back to the current (i.e., unchanged) context.

2.6 DISPLAY SCREEN TO GO TO AFTER A CONTEXT CHANGE

Issue: In response to a context change transaction initiated by the user via another application, some applications return to their top level screen upon a context change, while others remain on the current screen, and yet others go to a mid-level screen. This inconsistency causes confusion and sometimes frustration for users when trying to interact with multiple applications via a CCOW-enabled workstation.

Recommendation: To the fullest extent possible, applications should behave the same way whether they are responding to externally initiated context change transaction or to equivalent user inputs directed specifically at the application. Therefore an application should go to whatever screen it would go to if the user had explicitly selected via the application the object now in the context. In so doing, the application should also apply the same business, access privilege and security rules as it would had the user interacted directly with the application.

2.7 UNNECESSARY CONTEXT CHANGES

Issue: Some applications process all context change notifications that they receive, even if the new context is the same as the application's existing context. This can cause unnecessary performance delays.

Recommendation: Don't change context if the application is already tuned to the patient, user, etc. represented in the new context. Instead, just respond to the part of the context that has changed (e.g.,

observation changes but not patient). This requires that applications keep track of what they think is the current context so that they can compare and contrast with context change notifications.

2.8 EXPLICIT SUPPORT FOR BREAKING LINK

Issue: Some applications provide a control that enables users to explicitly break the application's context link at any time. This can result in user confusion when the control is accidentally used, or used without a complete understanding of what it means to break an application's link.

Recommendation: Support for any break link capabilities should be configurable via a single configuration item. (See section 2.9 for more details.)

2.9 BREAKING LINK WITHIN CONTEXT CHANGE DIALOG

Issue: The CCOW standard calls for allowing the user to break an application's link if one or more applications only conditionally accept a context change or if one or more applications are found to be busy during a context change. For some users the break-link capability can be confusing and results in unexpected application behaviors (e.g., not all of the user's applications "tune" to the same patient).

Recommendation: Application support for all break-link capabilities should be configurable via a single configuration item. This is the same item as described in section 2.8 above.

If the break-link configuration is enabled, then the explicit break-link actions described in 2.8, and the break-link option in the dialog shown when a conditional accept or busy application state arises should be presented by the application. If the break-link configuration is disabled, then the break-link action described in 2.8 should not be presented by the application and the break-link option should also be removed from the dialog shown when a conditional accept or busy application state arises.

When the break-link configuration is disabled and there is a busy application, the dialog should simply state that the selected action (change) cannot be completed (because of the busy application) and an "OK" button should be supplied to close the dialog.

An application's default configuration setting should be to have break-link disabled.

2.10 SAVING UNSAVED DATA

Issue: Some applications that are able to save unsaved data nevertheless still only conditionally accept context change transactions and as a result cause CCOW-related dialog box to be presented to the user. The frequent appearance of these dialog boxes can be annoying and/or confusing.

Recommendation: If an application has the ability to save unsaved data then it should do so and unconditionally accept when a context change transaction occurs. This will minimize the frequency at which CCOW dialog boxes are presented to users.

2.11 MODAL DIALOG BOXES

Issue: Some applications make substantial use of modal dialog boxes. These are commonly used to constrain user workflow. For example, some applications require the user to finish a modal task before he can interact with other parts of the application, and in so doing, may make prevent an application from being able to respond to a context change transaction initiated by another application. This runs counter to the CCOW paradigm, where an application is ideally ready to respond to an asynchronously received context change at any time.

Recommendation: Application developers should strive to design their applications in a way that avoids, or at least minimizes, the use of modal workflows so that the user is allowed to indirectly change the state (e.g., the patient it is "tuned" to) via interactions with other applications.

2.12 CONTROL OF THE DESKTOP

Issue: Some applications seek to control the desktop, especially by taking over the workstation's screen saver function and/or user inactivity timers. Some applications display their window full-screen and do not allow the window to be resized and/or minimized. This can create conflict with the user's other applications.

Recommendation: Applications should allow their windows to be resized or minimized in order to allow users to access the Microsoft Windows desktop start bar and other applications. Applications should not control the screensaver, and per the CCOW Standard, it must be possible for sites to disable, via a configuration setting, an application's control of inactivity timers. The timeout configuration default should be "no timeout".

2.13 SELECTIVELY ENABLING CONTEXT LINKS

Issue: Many applications implement multiple context links (e.g., user, patient, etc.), yet they do not allow this functionality to be individually enabled. Instead, all links are enabled, or none are. This makes it hard to implement CCOW systems where only a certain type of link is desired (e.g., only user link, or only patient link, etc.)

Recommendation: Applications should be configurable such that each context subject supported by the application can be enabled or disabled The default configuration should be that all context subjects supported by the application are enabled.

2.14 VALID USER ACCOUNTS AND PRIVILEGES

Issue: Some applications "tune" to the user presently in context and log the user on to the application, even if the user does not presently have a valid account. Some applications that have "tuned" to the user context allow the user to access data or functions within the application even when the user does not possess the necessary access privileges. Both of these situations generally arise when application support for CCOW User Link results in the creation of a different code path for logging on to the application wherein the new code path inadvertently bypasses protections afforded by the application's original logon code.

Recommendation: Applications should always verify that the user that it is about to automatically log on is a valid user. Applications should always enforce the same access privileges for a user who logs on via CCOW User Link as it would if the same user directly signed on to the application.

2.15 INACTIVITY TIMEOUTS

Issue: CCOW requires application inactivity timeouts to be configurable (including being disabled). Some applications do this on a global basis irrespective of whether the application is running in a context session or not. If the timeout is then globally configured to be off, there is no inactivity timeout when the application is used on a non-CCOW workstation or when its link with a context manager is broken.

Recommendation: The setting of the CCOW inactivity timer should only pertain to applications when they are joined to a context. When an application instance is running in stand-alone mode either because there is no context manager, because it cannot connect to the context manager, or because the application's link is broken then the setting of the configuration of the inactivity timer insofar as it applies to CCOW would not apply.

2.16 DOCUMENT ALL CCOW-RELATED CONFIGURATIONS

Issue: Applications are varied in terms of what CCOW capabilities can be configured. To the extent that some applications support CCOW configurations, they frequently make the configuration process harder than it should be for analysts to know what can be configured.

Recommendation: Applications should consolidate all of their allowed CCOW-related configurations in a single section or area of the application documentation.

2.17 PERFORMANCE ENHANCEMENTS

Issue: Applications do not always take advantage of various optimizations to improve performance.

Recommendation: Application designers should design applications to not create unnecessary processing and/or user actions overhead. Some things to consider are:

- Hidden or minimized applications – Upon receiving a context change notification, applications that are not visible may be able to avoid the overhead of refreshing their state or at least delaying the refresh until they become visible. There are various levels and techniques for optimizing here. In the most extreme case, an application that is often in a non-viewable state for long periods might suspend itself from the context and then resynchronize with the context when restored. (See section 3.5 for related information).
- Filtering – Applications that are not interested in a particular subject, should filter out that subject in order to reduce context change-related messaging overhead.

3 Common Mistakes

3.1 DISRUPTIVE CCOW-RELATED DIALOG BOXES

Issue: Some applications spuriously present CCOW-related dialog boxes to the user, which is disruptive to user workflow. For example, some applications ask the user if they want to object to a CCOW context change (i.e., conditionally accept) without a clear application need to object to or question the change. The best user experience is created when the need for CCOW-related dialog boxes is minimized.

Recommendation: Applications should minimize the need to conditionally accept context survey changes, which cause CCOW-related dialog boxes to be presented to the user. The application should have a good reason to object to the change, like the potential loss of uncommitted user-inputted data. Applications should make sure that the proposed context change actually affects it, as opposed to assuming that any and all changes require that the application only conditionally accept.

3.2 USE OF CCOW STATUS ICONS

Issue: Some applications change the appearance of the CCOW status icons from what is in the CCOW specification. Some applications use icons that look like the CCOW icons but are used for completely different purposes. Users get confused when multiple applications on their desktop have different-looking icons that mean the same thing, or similar-looking icons that mean different things.

Recommendation: Applications should use the CCOW status icons as defined in the CCOW specification. Applications should not use icons that look similar to CCOW Status icons as this could be confusing.

3.3 UNINFORMATIVE CONTEXT CHANGE CONDITIONAL ACCEPT MESSAGES

Issue: Not all applications provide an informative message when they conditionally accept. Some applications do not provide any message. This results in user confusion and uncertainty about what to do in a conditional accept situation.

Recommendation: Applications should always provide a reason for their conditional accept. These messages need to be informative. The UIS calls for a 63 byte limitation on the length of these messages. This presents a challenge in terms of just how informative conditional accept messages can be. Therefore application developers need to be particularly thoughtful in terms of what they say in these messages. In addition, a meaningful application name specified in the JoinCommonContext method should be used so that the user can more easily determine the application that generated the conditional accept message.

3.4 GRABBING WINDOW FOCUS

Issue: In responding to a context change some applications take the window focus away from the active application (i.e. the application the user was working with when they initiated the context change). This creates problems for the user who is not expecting the window focus to change.

Recommendation: An application should ensure that it does not take focus from another application when responding to a context change. Some implementation frameworks (e.g. applets running under certain JVMs) grab focus automatically. Under such circumstances, explicit measures may be needed to conform to this recommendation.

3.5 DISPLAYING CONTEXT SENSITIVE INFORMATION WHEN MINIMIZED

Issue: Some applications display context-sensitive information even when minimized, for example they show the patient name in the label they present in the Microsoft taskbar.

Recommendation: If an application is minimized and the application's icon displays context sensitive information (e.g., a patient's name), then the application must be sure that this information remains current with the context. (See section 2.17 for related information.)

3.6 MAXIMIZING WHEN RESPONDING TO A CONTEXT CHANGE

Issue: Some applications, when minimized, maximize when responding to a context change event.

Recommendation: A minimized application should remain minimized even when it participates in a context change.

3.7 IMPROPER WINDOW SIZING

Issue: Some applications do not properly size their windows, either resulting in windows that clip windows from other applications, do not make room for other windows from other applications, or do not resize properly. This results in user frustration when using multiple applications. Examples include windows that occupy the full screen even though other windows have priority, such as the Windows taskbar, and applications whose display content is clipped because a window is not properly sized.

Recommendation: Applications need to respect the GUI standards for the target technology used to implement the application (e.g., Windows, Swing, etc.) so that they size and resize their windows properly.

3.8 INTERACTING WITH THE USER DURING A CONTEXT CHANGE TRANSACTION

Issue: Some applications accept a CCOW context change transaction initiated by another transaction, but then attempt to interact with the user by popping up a dialog box. Any attempt to interact with the user by an application that responds to a context change is inconsistent with the CCOW specification and is problematic to the user workflow. When an application displays status or warning information in a dialog box, especially one that is modal, it can create a stall point in the user's workflow that may not be apparent to the user.

Recommendation: Applications should not present dialogs boxes (especially modal dialog boxes) when responding to a context change transaction initiated by another application.

3.9 CRASHING DUE TO UNEXPECTED CONTEXT DATA ITEM NAMES

Issue: Some applications display an error message or crash when they receive context data whose item names (including suffixes) are not among the names of the items that are coded within the application.

Recommendation: Per the CMS, applications must gracefully be able to receive one or more context data items whose names it does not recognize or does not expect (e.g., an application that only support user and patient context is presented with encounter context data as well) without creating an error state and without alerting the user to the circumstance. These circumstances must not be communicated through the context change survey response.

3.10 CRASHING DUE TO UNEXPECTED CONTEXT DATA ITEM VALUES

Issue: Some applications display an error message or crash when they receive context data items whose values they do not understand or cannot process.

Recommendation: Per the CMS, applications must gracefully be able to receive one or more context data items whose values it does not recognize or does not expect (e.g., a medical record number not known to the application) without creating an error state and without alerting the user to the circumstance. These states must not be communicated through the context change survey response. When this occurs the application should not display context-sensitive data for the item type.

3.11 SECURE BINDING

Issue: Some applications that access secure context subjects, such as the User subject, either read the context without performing a secure binding with the context manager, or they do perform a secure binding but they do not validate the digital signatures that they subsequently receive from the context manager. In either case it is not possible for the application to ascertain that the secure subject context data that it gets has indeed come from a valid context manager.

Recommendation: Applications should not get the data for a secure context subject if a secure binding has not been performed. Applications should always validate the digital signatures that it receives before assuming that the signed context data is valid.

3.12 LEAVING THE CONTEXT WHEN TERMINATED

Issue: Some applications do not leave the CCOW context when they are terminated. This is frequently true of web applications.

Recommendation: When an application is instructed to terminate, it must leave the CCOW context.

3.13 CONFIGURABLE SUFFIX

Issue: Some applications do not allow their context data item names to have configurable suffixes.

Recommendation: The use of configurable suffixes is required in order be compliant with the CCOW standard. Further, it is recommended that the entire subject name used by each application for each subject that it supports be configurable.

3.14 APPLICATION LOG OFF VS. CONTEXT LOG OFF

Issue: Some applications clear part or all of the context when the user terminates or logs off of the application. This causes unexpected situations, especially if the user was intending to continue to use the other applications that are still participants in the context session (e.g., these applications are put into a no-user and/or no-patient, etc., state).

Recommendation: Applications should be configurable so that it is possible to control what happens to the common context data when they log off or terminate an application that is a participant in the context. The default configuration should be that logging off of the application, or terminating the application, only affects the application and the common context data should not be affected by the application when it leaves the context. The other configuration option is that the user subject is set to empty when the user logs off of or terminates the application. This has the effect of logging the user off of the common context session. In this case the other context data should not be altered by the application when the user logs off or terminates. This recommendation pertains to clinical style applications and not applications functioning as user or desktop manager. The default configuration setting should be to log off of just the application.