1 Security Considerations

This specification considers two sets of security requirements, those of the applications that use the WS-Eventing protocol and those of the protocol itself.

This specification makes no assumptions about the security requirements of the applications that use WS-Eventing. However, once those requirements have been satisfied within a given operational context, the addition of WS-Eventing to this operational context must not undermine the fulfillment of those requirements; the use of WS-Eventing should not create additional attack vectors within an otherwise secure system.

There are many other security concerns that one may need to consider when implementing or using this protocol. The material below should not be considered as a "check list". Implementers and users of this protocol are urged to perform a security analysis to determine their particular threat profile and the appropriate responses to those threats.

1.1 Notifications

The information contained in Notifications might be sensitive. In such cases it is advisable to authenticate subscribers and perform the appropriate authorization checks as part of the processing of the Subscribe request. Note that some Notification streams might require per-Notification authorization checks. This could be necessary in cases where the sensitivity of the Notification information is not known at Subscribe time or might vary over the lifetime of a subscription.

As with any message sent over a network, Notifications should have integrity and confidentiality protections applied in proportion to the sensitivity of the information they contain.

The ability to subscribe on behalf of a third-party Event Sink could be misused to create a denial-of-service attack. While it does not remove the ability for such misuse, authenticating subscribers provides a way to trace the origin of such attempts. Additionally, the authorization of subscribers reduces the pool of potential attackers.

1.2 Subscriptions

Once created, subscriptions to sensitive Notification streams should be treated as protected resources. Renew, GetStatus, and Unsbuscribe requests should be authenticated and the identify of the requester should be checked against the “sink owner” of the subscription (for example, the entity that performed the original Subscribe request). Likewise SubscriptionEnd messages should be authenticated and verified to originate from the “source owner” of the subscription (for example, the entity that sent the original SubscribeResponse message). Note that determinations of subscription ownership are particular to individual deployments. For example, within a particular organization it might be the rule that “source owner” of a subscription is considered to be the Event Source (i.e. the entity that processes the original Subscribe request) of that subscription.

1.3 Endpoint Verification

Implementations that perform validity checks on the EPRs used in WS-Eventing (wse:NotifyTo, wse:EndTo) should be aware that such checks can be misused to obtain

information about a target network. For example, suppose an Event Sink implementation verifies the address of NotifyTo EPRs by attempting to create a connection to this address and faulting the Subscribe request if such a connection cannot be created. When deployed within a DMZ, such an Event Sink could be exploited to probe for other, non-visible machines within the DMZ by guessing target address values and using these values in Subscribe requests. Note that, even if the returned fault does not provide enough information to enable such attacks, the time the Event Sink spends waiting for a connection timeout (or not) may betray the existence or non-existence of a host at the target address.

Implementations that perform validity checks on the EPRs used in WS-Eventing should provide a way for such checks to be disabled in environments where these types of attacks are an issue.