Table of Contents

Programmers Reference, Methods, IChannelSettingsEditor, IChannelSettingsEditor Methods, ValidateSettings

validateSettings (Properties, int)

```
public void validateSettings (Properties settings, int intelligenceType)
throws Exception
```

This method is called when changes to the business object (which include changes to the channels) are committed. The editor is supposed to validate the settings and throw the exception with the appropriate message if the settings are invalid. The validation should happen during editing as well, however, this method may be useful when the editor was never invoked by the user in the first place so that the default settings are committed. The default settings for the channel, however, may not be complete and need validation.

Parameters

`intelligenceType`	the intelligence type of the business object for which the channel is configured. Possible values are specified in the getIntelligenceTypeApplicability method of the `IChannelType` interface.
`settings`	the current values of the channel settings to be validated.

Throws

The exception if channel settings are invalid.

> **note**
>
> If you want **Aware IM** to build the channel with the intelligent business object when the **Aware IM** server starts up so that **Aware IM** receives notifications from the business object, set the value of `CommonConstants.CHANNEL_SETTINGS_BUILD_NOTIF_CHANNEL` to be "true" in the channel settings.

Below is an example of the Channel Settings Editor that edits the settings of the E-mail channel:

```
public class EmailSettingsEditor implements IChannelSettingsEditor
{
    public EmailSettingsEditor()
    {}
    public PropertiesstartEditing (int intelligenceType, Componentparent,
PropertiesinitialSettings, boolean viewOnly)
    throws Exception
    {
        StringhostServer =null;
        StringfromAddress=null;
        StringuserName=null, password=null;
        if (initialSettings!=null)
        {
```

```java
            hostServer =initialSettings.getProperty (MAIL_HOST);
            fromAddress=initialSettings.getProperty (MAIL_FROM_ADDRESS);
            userName =initialSettings.getProperty
(CHANNEL_SETTINGS_USER_NAME);
            password =initialSettings.getProperty
(CHANNEL_SETTINGS_PASSWORD);
        }
        EmailSettingsDlgdlg=null;
        if (parent instanceof Frame)
        {
            dlg=new EmailSettingsDlg((Frame)parent,hostServer,
fromAddress,userName,password,viewOnly);
        }
        else if (parent instanceof Dialog)
        {
            dlg= new EmailSettingsDlg((Dialog)parent,hostServer,
fromAddress,userName, password,viewOnly);
        }
        dlg.show();
        if (dlg.getReturnStatus()!=EmailSettingsDlg.RET_OK)
            return null ;
        Propertiesprops=new Properties();
        props.setProperty(MAIL_HOST,dlg.getHostName());
        props.setProperty(MAIL_FROM_ADDRESS,dlg.getFromAddress());
        props.setProperty(CHANNEL_SETTINGS_USER_NAME, dlg.getUserName());
        props.setProperty(CHANNEL_SETTINGS_PASSWORD, dlg.getPassword());
        return props;
    }
    public void validateSettings(Propertiessettings, int intelligenceType)
    throws Exception
    {
        StringhostServer = null ;
        StringfromAddress= null ;
        if (settings!= null )
        {
            hostServer =settings.getProperty(MAIL_HOST);
            fromAddress=settings.getProperty(MAIL_FROM_ADDRESS);
        }
        if (hostServer== null ||hostServer.length()==0)
            throw  new Exception("Mail host name for e-mail channel is not
specified");
        if (fromAddress== null ||fromAddress.length()==0)
            throw  new Exception("Mail 'from' address for e-mail channel is
not specified");
    }
}
```