```
/***
    USAGE:

    DependencyErrors(components, license, permissions)

    PARAMETERS:

    components : list of maps, [{}, {}, {}, ...]
        Components to test. Each component defines
            name => component name or capability to test for
            sid => Service ID (sid) to check for in license. If present,
does a service check.
            requires => Capability to check in the license. If present,
does a capability check.
            (optional) title => User-friendly component name
            (optional) link => Link for more information in case of error

    (optional) license : str (default: nil)
        Site license type to check for; one of 'commercial' or nil

    (optional) permission : str (default: nil)
        Permissions required for current user; one of 'admin' or nil

    VERSIONS:

    1.0     15-Jun-10    kalida     initial version
    1.1     15-Jul-10    kalida     added capability checks
    1.2     03-Mar-11    kalida     updated regex for bug MT-9851

***/

var components = $0 ?? $components ?? [];
var license = $1 ?? $license ?? nil;
var permission = $2 ?? $permission ?? nil;

var licenseError = nil;
var permissionError = nil;
var componentErrors = nil;

// general data lookups
var licenseXML = wiki.api(site.api & 'license');

// if no explicit expiration, use a far-future date
var licenseExpires = xml.date(licenseXML, 'date.expiration') ??
date.addyears(date.now, 100);
var prefix = 'mindtouch.templates.dependencyerrors.';
var sidList = [];

// gather existing licensed services
```

```
foreach (var serviceLicense in licenseXML['grants/service-license']) {
    var sid = string.Match(xml.text(serviceLicense),
'(?<=sid=")(.*?)(?=")');

    // if valid sid, add with expiration date
    if (#sid > 0) {
        var sidExpires = licenseExpires;

        // attempt to find expiration date in text or as attribute
        var expires = string.Match(xml.text(serviceLicense),
'(?<=expire=")(.*?)(?=")');

        if (#expires > 0) {
            let sidExpires = date.parse(expires);
        } else if (#xml.text(serviceLicense, '@date.expire') > 0) {
            let sidExpires = xml.date(serviceLicense, '@date.expire');
        }

        let sidList = sidList .. [{sid: sid, expires: sidExpires}];
    }
}

// gather licensed capabilities
var capabilityGrants = {};
foreach (var grant in
licenseXML['grants/*[not(name()="service-license")]']) {
    var grantName = xml.name(grant);
    var grantValue = xml.text(grant);
    let capabilityGrants = capabilityGrants .. { (grantName): grantValue};
}

// check license
if (license == 'commercial') {
    var currentLicense = xml.text(licenseXML, '@type');

    if (license != currentLicense && currentLicense != 'trial') {
        let licenseError = wiki.localize(prefix .. 'error.license');
    }
    else if (date.Compare(date.now, licenseExpires) > 0 ) {
        let licenseError = wiki.localize(prefix ..
'error.license.expired');
    }
}

// check user permissions
if (permission == 'admin') {
    if (!user.admin) {
        let permissionError = wiki.localize(prefix .. 'error.permission');
    }
}
```

```
// check components
foreach (var component in components) {
    var errorMessage = nil;
    var title = component.title ?? component.name;
    var link = component.link ?? 'http://www.mindtouch.com/redir/
dependency/';

    // if component.requires available is a cabality check
    var serviceCheck = component.requires is nil;

    // admins are able to check if service licensed
    if (user.admin && component.sid && serviceCheck) {

        // component is valid if any licensed sid is a prefix match
        var foundLicense = false;
        var expiredLicense = false;
        foreach (var sidItem in sidList) {
            if (string.StartsWith(component.sid, sidItem.sid)) {
                let foundLicense = true;

                if (date.Compare(date.now, sidItem.expires) > 0 ) {
                    let expiredLicense = true;
                }

                // found component, avoid searching others
                break;
            }
        }

        if (!foundLicense) {
            let errorMessage = <a href=(link)
rel="component-unlicensed">wiki.localize(prefix ..
'error.component.unlicensed', [title])</a>
        } else if (expiredLicense) {
            let errorMessage = <a href=(link)
rel="component-expired">wiki.localize(prefix ..
'error.component.expired', [title])</a>
        }
    }

    // name is required
    if (!component.name) {
        let errorMessage = wiki.localize(prefix ..
'error.component.unnamed');
    }
    else if (!errorMessage && !__env[component.name] && serviceCheck) {

        // @TODO kalida: in future, have separate checks for not
installed vs. not running
```

```
        let errorMessage = <a href=(link)
rel="component-missing">wiki.localize(prefix ..
'error.component.missing', [title])</a>
    }

    // if capability check, ensure it is granted
    if (!serviceCheck && capabilityGrants[component.name] !=
component.requires) {
        var currentValue = capabilityGrants[component.name] ??
wiki.localize(prefix .. 'error.capability.missing.default');
        let errorMessage = <a href=(link) rel="capability-missing">
            wiki.localize(prefix .. 'error.capability.missing', [title,
component.requires, currentValue])
            </a>;
    }

    // add this error to the list
    if (errorMessage) {
        let componentErrors = componentErrors .. [{component: component,
message: errorMessage}];
    }
}

// list missing dependencies
if (licenseError) {
    <div class="error-dependency error-license">
        licenseError;
    </div>
} else if (permissionError) {
    <div class="error-dependency error-permission">
        permissionError;
    </div>
} else if (componentErrors) {
    <div class="error-dependency error-component">
        wiki.localize(prefix .. 'label.components.error');
        <ul>
            foreach (var error in componentErrors) {
                <li class=(error.component.name)>
                    error.message;
                </li>
            }
        </ul>
    </div>
}
```